GEORGIA INSTITUTE OF TECHNOLOGY — OFFICE OF CONTRACT ADMINISTRATION

PROJECT ADMINISTRATION DATA SHEET

[X] ORIGINAL [ ] REVISION NO. ____

Project No. E-25-633 R6086-0A0 GTRC/GIT DATE 2 / 7 / 86

Project Director: Dr. Robert Fulton School/~~XXX~~ MECH ENG

Sponsor: General Motors Corporation Warren, Michigan 48090-9055

Type Agreement: Research Project Agreement Dated 1/21/86

Award Period: From 1/1/86 To ~~8/31/86~~ 8/31/88 (Performance) 12/31/86 (Reports)

Sponsor Amount:	This Change	Total to Date
Estimated: $		$
Funded: $	49,635.00	$ 49,635.00

Cost Sharing Amount: $ Cost Sharing No:

Title: Computational Crash Dynamics Methods for Fifth Generation Supercomputers

ADMINISTRATIVE DATA OCA Contact R. Dennis Farmer X4820

1) Sponsor Technical Contact:

Dr. James Bennett
Assistant Department Head
Engineering Mechanics Department
General Motors Research Laboratories
Warren, MI 48090-9055
(313) 575-2927

2) Sponsor Admin/Contractual Matters:

Mr. Robert F. Schultz
Executive Administrator
General Motors Research Laboratories
Warren, MI 48090-9055

Defense Priority Rating: Military Security Classification:
(or) Company/Industrial Proprietary:

RESTRICTIONS

See Attached ____ Supplemental Information Sheet for Additional Requirements.

Travel: Foreign travel must have prior approval – Contact OCA in each case. Domestic travel requires sponsor approval where total will exceed greater of $500 or 125% of approved proposal budget category.

Equipment: Title vests with NONE PROPOSED

COMMENTS:

COPIES TO:

SPONSOR'S I. D. NO. 02.206.000.86.004

Project Director	Procurement/GTRI Supply Services	GTRC
Research Administrative Network	Research Security Services	Library
Research Property Management	Reports Coordinator (OCA)	Project File 2/7/86
Accounting	Research Communications (2)	Other Jones/Legal

FORM OCA 65:285

12 / 88

GEORGIA INSTITUTE OF TECHNOLOGY
OFFICE OF CONTRACT ADMINISTRATION

NOTICE OF PROJECT CLOSEOUT

Date 12/9/88

oject No. E-25-633 Center No. R6086-0A0

oject Director R.E. Fulton School/Lab ME

onsor General Motors Research Lab

ntract/Grant No. 15-8 GTRC XX GIT ____

me Contract No. N/A

tle Computational Crash Dynamics Methods for Fifth Generation Supercomputers

fective Completion Date 8/31/88 (Performance) 8/31/88 (Reports)

oseout Actions Required:

_ None
_ Final Invoice or Copy of Last Invoice
_ Final Report of Inventions and/or Subcontracts Patent Questionnaire sent to P/I.
_ Government Property Inventory & Related Certificate
_ Classified Material Certificate
_ Release and Assignment
_ Other _____

cludes Subproject No(s). N/A

project Under Main Project No. N/A

ntinues Project No. _____ Continued by Project No. _____

tribution:

_ Project Director
_ Administrative Network
_ Accounting
_ Procurement/GTRI Supply Services
_ Research Property Management
_ Research Security Services

x Reports Coordinator (OCA)
x GTRC
x Project File
x Contract Support Division (OCA)
___ Other _____

COMPUTATIONAL CRASH DYNAMICS METHODS FOR FIFTH GENERATION SUPERCOMPUTERS

Robert E. Fulton
Kuo-Ning Chiang

George W. Woodruff School of Mechanical Engineering
Georgia Institute of Technology
Atlanta, GA 30332

November 1988

Abstract

In the area of crash impact, research is critically needed on the development and evaluation of parallel methods for crash dynamic analysis of complex nonlinear finite element and/or finite difference structure problems. The development of general-purpose finite element structural analysis computer programs have provided the capability to address a wide range of structures problems. However, these software systems are severely limited for nonlinear transient calculations because of the available speed on current sequential computers.

Projected advances in computer technology indicate that significant increases in effective calculation speed will available in the 1990's, through fifth generation supercomputer architectures consisting of arrays of processors operating in parallel on different tasks (see e.g., ref. 1 for a survey). Such advanced supercomputers, denoted as MIMD (multiple instruction, multiple data) computers, have the potential for increasing effective calculation speeds by several orders of magnitude. But this potential increase in speed can not be effectively utilized without the development and implementation of appropriate numerical algorithms for structures which take advantage of the parallel computation features of this new generation of computers. Use of existing conventional algorithms and software will not realize the full potential of these new MIMD computers, and research is needed in the development of parallel structural analysis/design algorithms for these computers. This research has been to develop and evaluate parallel methods for crash dynamic analyses of complex nonlinear finite element and finite difference structure problems.

Table of Contents

1. Introduction 1

2. Literature Survey 3

- •2.1 Parallel Computers 3
- •2.2 Relevant Progress In Parallel Hardware 4
- •2.3 Parallel Algorithms 7
- •2.4 Crash Dynamics 9

3. Fundamental Concept for Solid Mechanics 12

- •3.1 The Strains 12
- •3.2 The Total Lagrangian Description 13
- •3.3 General Derivation of F.E. Equilibrium Equation 15

4. Nonlinear Dynamic Algorithm 17

- •4.1 FENRIS - General Purpose Nonlinear FES 17
- •4.2 Nonlinear Dynamics Equilibrium Formulation 17
- •4.3 Implicit Time Integration Algorithm 19
- •4.4 Iterative Schemes 21
- •4.5 CPU and Storage 22

5. Application to FEM Analyses 25

- •5.1 Sequential Verification 25
- •5.2 Static Analysis 25
- •5.3 Crash Dynamic Analysis 26

6. The Parallel Element Generation Theory 28

- •6.1 Element Formulation 28
- •6.2 Parallel Element Matrix Generation 30

•6.3 Hybird Parallelism 31
•6.4 Subdomain Parallelism 32
•6.5 Merge of System Stiffness Matrix 32
•6.6 Concluding Remarks 33
7. Internal Force Vector Generation 35
•7.1 Concluding Remarks 36
8. Summary 37
• Reference 39
• Table 45
• Figure 48

1. Introduction

The finite element method is widely used as a computational method to model physical system in various engineering problems. For detailed analyses of complex designs, structural models composed of thousands of degrees of freedom are no longer uncommon. Parallel computers offer the promise for solution to complex problems in detail considered too time-consuming for todays sequential processors (ref.1). To benefit from advances in parallel computers, software must be developed which take maximum advantage of parallel processing features.

The parallel implementation criteria that influence the efficiency of an algorithm include the amount of computation versus the number of processors, the communication paths, waiting and synchronization delays, critical regions of the algorithm that must be executed sequentially and the the size of a problem in relation to the number of processors used.

Finite element analysis and optimization of structures subjected a static or dynamic forces typically require the solution of large systems of linear equations, element generation and force generation. Many commercial finite element codes (ref. 2-4) use a triangular decomposition of the system matrix in combination with a forward/backward substitution to solve the global equilibrium equations. The decomposition of the system may contribute significantly to the computing cost. For example in the static stress analysis it may take more than 50% of the total execution time (ref. 5). Recently the parallel matrix decomposition of the system matrix decomposition has been investigated by several researcher (eg ref. 6-9) but the proposed solutions do not readily fit into the environment of a multi-purpose finite element code. A parallel finite element method

equation solver has been developed in the present study and tested for several static and dynamic stress analysis demonstration problems. It has also been incorporated in a production finite element system (FENRIS) and applied to S-Frame torque beam crash test problems. The results (GM Project Final Report Phase II, ref. 10 and 11) indicate that the significant speedup can be achieved through the use of many processors for an appropriate finite element problems. Also from the above observations one finds that the nonlinear force generation and element stiffness generation are extremely important in the area of nonlinear transient analyses and research is critically needed on the development and evaluation of parallel methods for element generation and nonlinear force generation.

This research is a continuation and culmination of research on parallel crash dynamic methods (see ref. 10) and has focused on the investigation of the following three areas

1. Element generation.
2. Nonlinear forces generation.
3. Global-Local data mapping

The following sections address each of these areas. Section 2, 3 and 4 produce a survey of relevant literature, solid mechanics theory and nonlinear dynamics concepts, respectively. Section 5 shows some applications to FEM analysis and parallel performance. Section 6 cover the first and third area, Section 7 cover the second and third area. Specific summaries or conclusions relative to each of these areas is included at the end of the respective sections.

2. Literature Survey

2.1 Parallel Computers

The impact of supercomputers, and highly parallel processing systems on finite element computation is profound. This topic has been reviewed in detail by Noor, Storaasli, and Fulton (ref. 1). Parallel computers can be classified as either SIMD (single instruction multiple data) or MIMD (multiple instruction multiple data) architectures. In the class of SIMD are such machines as the CDC STAR-100, CYBER 203, CYBER 205 and CRAY-1; these provide high computation speed, but are generally not well suited for large-scale finite element computations. Complex structural dynamics problems will require effective computer speeds much greater than 10^3 MFLOPS (million floating point operations per second) for timely results (ref. 1), and these speeds are well beyond the capabilities of current SIMD computers. A more effective way of increasing computation speeds is through the use of multiprocessor computers. MIMD computers can be classified by their memory arrangement and the means by which each processor can access data stored in memory. In this classification there are two types of architectures; (1) shared memory architectures such as the CRAY X-MP, IBM 3090, CRAY-2, ETA-10, ALLIANT FX/8 and FLEX/32, and (2) network or local memory message based architectures such as the FPS T-series, iPSC INTEL, N-CUBE, AMETEK, and Caltech MARK III Hypercube.

2.2 Relevant Progress In Parallel Hardware

Significant advances have taken place over the past year in parallel/vector supercomputer hardware which can provide directions relative to finite element computations and appropriate software. It is useful to classify the evolving supercomputer family into four categories, moderately parallel high speed system, massively parallel system, moderately parallel intermediate systems, and high performance parallel work stations. Table 1 gives a list of the parallel/vector computers in the four classes.

Table 1 shows that at one extreme are the high speed moderately parallel systems such as CRAY and IBM systems. At the other extreme are the massively parallel systems such as hypercube or butterfly systems. In between are various intermediate systems such as Alliant and Convex system. The fourth group corresponds to high performance parallel/vector workstations, a number of which were introduced at SIGGRAPH 88 (August/88). This latter group indicates the evolution of the established workstation market toward the new technology of parallel/vector processing; Table 1 gives data on speed and memory for these systems.

During the past year several events have occured which give an indication of the opportunities for FEM software on commercial parallel/vector computers. For example, supercomputer conferences were held in the U.S., Norway and Japan (Table 2). Especially of interest is the noticeable change in IBM's posture relative to parallel computers. Until recently IBM largely ignored parallel processing but IBM is now marketing the IBM 3090 as a parallel computer with 6 processors. Steve Chen, formerly of CRAY, now leads an IBM design team rumored to produce competition for CRAY. IBM's

Bergen Scientific Center sponsored the Norway conference focused largely on parallel technology. These and other IBM actions make it clear that IBM now considers parallel processing to be a significant market and their action makes parallel computing credible. The IBM action will quickly influence the Japanese supercomputing market which to date has concentrated only on very high performance vector processors.

Finally CRAY has announced its plans for the future (Table 3) which indicate a growing level of parallelism to go with its vector and large memory capability. The Pittsburg Supercomputer Center has already committed to obtaining the first 16 processor CRAY-3 to be delivered in 1989 and CRAY projects a 64 processor CRAY-4 in 1992.

These results lead to several significant conclusions relative to the future of parallel processing.

1. The moderately parallel large scale computer are an established applications market with CRAY now selling 50% of its computers to industrial organizations. Furthermore, CRAY is moving steadily toward increasing numbers of parallel processors. The ETA and IBM activities make it a competitive market. To date the Japanese (Hitachi, Fujitsu and NEC) have focused on maximizing vector capabilities but the IBM strategies and the Tokyo Conference indicate that Japan will soon expand to parallel architectures.

2. The massively parallel computer market (e.g. Hypercube, Butterfly, etc.) is not yet well established and usages are basically research oriented (some of it FEM

oriented). Interest is still growing but the absence of general purpose FEM software for such machines inhibits their practical use. Nevertheless there is growing set of special purpose software; some eye catching FEM results have begun to evolve (e.g. Sandia, Table 2); and the vendors are marketing aggresively

3. The moderately parallel intermediate system market (e.g. Alliant, Convex) is still evolving and these systems are being used as less expensive alternatives to the CRAY or as CRAY frontends. Informal projections are that Alliant will soon move to a 16 or 32 processor capability.

4. The recent explosion on the scene (Table 1) of a large number of moderately parallel/vector based high performance workstations is an especially important event and is indication of the future. Both the Ardent and Raster systems have parallel and vector capability and Apollo and Raster have rated performance exceeding 100 MFLOPS. Such systems provide a parallel/vector graphics and computation capability on the engineering desk. These trends suggest that high performance parallel workstations will soon be highly competitive with the parallel intermediate systems such as Alliant and Convex. This would be similar to that which occured when workstations began to take over much of the minicomputer market for sequential processors.

The results indicate the continued evolution of the massively parallel computer market. This may occur through the acceptance of innovative minicomputer systems or by attached accelerators based on the hypercube or Butterfly configurations. The

massively parallel approach for the high speed systems also appears to be steadily occuring through the continued growth of the CRAY type general purpose computers, with CRAY planning at least 64 processors in 3-4 years. IBM is also known to be studying massively parallel architectures. These results strongly indicated that the hardware base will evolve for large number of parallel/vector processors. Commercial finite element systems in 3-5 years will be needed which run effectively on 100-1000 processors.

2.3 Parallel Algorithms

In adapting a solution to parallel processing implementation, efficient algorithms typically maximize parallel and minimize sequential calculation. The algorithm must consider such things as: efficient interaction of vector and parallel computations, communication/waiting overhead, memory contention, critical regions of the code that must be executed sequentially, data interdependency, idle time resulting from imbalance of the workload, portability and reliability. In general, for concurrent transient finite element systems two alternate strategies should be considered, one based on an explicit method and the other based on an implicit method. An obvious way to carry out a concurrent execution of explicit finite element method using n processors is to decompose the structural domain into n regions and give each processor "responsibility" for solving an entire subproblem. A substructuring method for implementation on multiprocessor computers has been recently proposed by Storaasli and Bergan (ref. 47). An automated mesh decomposition and concurrent finite element analysis has been discussed by Malone (ref. 19). His approach used a central difference time integration scheme and selected test problems were implemented on Hypercube multiprocessor computers. The formulation includes a new decomposition algorithm which automatically divides an arbitrary finite element mesh into region and assigns each region to a

processor on the Hypercube, The algorithm has been implemented on a 32 processor Hypercube for plane-stress analysis and exhibits more than 95% efficient use of machine. The Newmark trapezoidal integration approach has been carried out by Storaasli, Ransom and Fulton (ref. 27), and their results for a two dimensional beam grillage problem gave computation speedups reaching a value of 6.5 for eight processors.

When the time integration procedure uses a direct approach, a key computationally intensive step is the decomposition of the symmetric/unsymmetric system matrix. The Cholesky decomposition (a scheme widely used in the existing finite element equation solver) has been studied by Goehlich, Komzsik and Fulton (ref. 40). Their parallel approach was incorporated in MSC/Nastran and tested for several static stress analysis demonstration problems on CRAY X-MP and IBM 3090 computers with up to four processors. The results indicated that a parallel processing approach can significantly reduce execution time for large scale finite element problems. A parallel Cholesky scheme is also discussed by Bostic and Fulton (ref. 29) in their Lanczos method implementations. In the Lanczos method the decomposition step is the most time-consuming calculation step for large problems and this step benefited the most from the parallel procedure. A speedup of 7.8 on eight processors was obtained for the decomposition step of the mast problem on the FLEX/32 Multicomputer.

It is well known that local memory machines have difficulty in achieving a good speedup for algorithms such as the Cholesky decomposition. The sequential Frontal method was first described by Irons (ref. 51) to solve finite element problems, and is based on the observation that Gaussian elimination can begin before assembly of the K matrix is completed. In fact a node in the finite element mesh can be eliminated as soon

as all the elements associated with it are assembled. A parallel multifrontal equation solver was investigated by Geist (ref. 41) using a local memory multiprocessor (iPSC Hypercube). The approach is to have one front on each processor and is designed so that each processor solves an entire subproblem. While the limited results are encouraging it requires regeneration of the K matrix in every iteration, a strategy not well suited for a general purpose nonlinear finite element analysis.

An alternative approach is the conjugate gradient iteration equation solver, a method studied by Lyzenga, et al (ref. 45) on a 32 processor Caltech Hypercube. It was used successfully for a large number of finite element problems. The conjugate gradient algorithm is well-suited for vector computation but, because of its many synchronization points, it more difficult for parallel computation. A block conjugate gradient algorithm has been discussed by O'Leary (ref. 65), and the parallel efficiency of his algorithm exceeded that of the standard conjugate gradient algorithm. The block preconditioned conjugate gradient method has also been investigated by Meurant (ref. 68) on a four processors computer CRAY X-MP/48, where the numerical results exhibit good parallel and vector efficiency. Other studies of parallel equation solvers have focused on narrow banded systems using such methods as cyclic reduction, recusive doubling, divide and conquer and twisted factorization methods (ref. 55, 66 and 67). The results show a good speedup potential in both parallel and vector computations.

2.4 Crash Dynamics

The associated tasks for crashworthiness are: prediction of the crashworthiness of a specific structure, improvement in design, and possibly optimization of the structure with respect to crashworthiness. In the past, the evaluation of crash performance was dependent on the impact tests, but such tests are costly and require a long preparation

time (ref. 42, 53). Therefore, numerical studies of crash analyses are becoming increasingly important in this area. For a complicated crash simulation, the amount of data required to describe the problem is enormous, and results in extremely large computational costs. One of the first finite element programs to take full advantage of vector processes, such as the CRAY-1, is DYNA3D (ref. 64, 69). The DYNA3D system has been reported by Benson, Hallquist and Stillman (ref. 69). One of the test problem shown at Figure 1, is a curved S-Frame impacted by a large mass at 30 km/hour. The algorithm used for the dynamic analysis was the explicit central difference method. The finite element model consists of 1600 shell elements with five integration points through the thickness. The computation for 35 ms of real time required nearly 4.2 CPU hours on the CRAY-1.

Computer simulation of crash phenomena has also been studied by Argyris, Balmer and Kurz (ref. 46) using the S-Frame test problem (see Figure 2). Their finite element model of the S-Frame is composed of 388 TRUMP element with 1200 unknowns, and the dynamic computation was performed via a matrix solution of Newmark's method (ref. 15). The time increment was 0.0005 sec and the number of iterations per time step was limited to 5. Computation up to 45 ms (90 time intervals) required 1.5 CPU hours on the CRAY-1M (the estimated CPU-time on VAX-11/780 was 300 hours).

The results obtained do not adequately fit the experimental result, and a refined mesh and smaller time step is required to obtain better results. The results discussed here show that todays supercomputers are marginal at best for sophisticated crashworthiness calculation. This implies that the solution algorithm must not only be inherently efficient, but also that it must take advantage of both vector and parallel computers so that

all of a supercomputer's potential is realized. In the area of crash impact, research is critically need on the development and evaluation of vector and parallel methods for crash dynamic analysis of complex nonlinear transient finite element problems.

3. Fundamental Concept for Solid Mechanics

3.1 The Strain

Consider an infinitesimal line element connecting the point A (X_1, X_2, X_3) Figure 3 to a neighboring point A' $(X_1 + dX_1, X_2 + dX_2, X_3 + dX_3)$ The square of the length ds_o of AA' in the original configuration is given by

$$ds_o^2 = \delta_{ij}\, dX_i dX_j \tag{3.1}$$

When A and A' are deformed to the points B (x_1, x_2, x_3) and B' $(x_1 + dx_1, x_2 + dx_2, x_3 + dx_3)$, the square of the length ds of the new element BB' is

$$ds^2 = \delta_{ij}\, dx_i dx_j \tag{3.2}$$

The difference between the squares of the length may be written as

$$ds^2 - ds_o^2 = (\delta_{mn} \frac{\partial x_m}{\partial X_i} \frac{\partial x_n}{\partial X_j} - \delta_{ij})\, dX_i dX_j \tag{3.3}$$

where δ is the Kronecker delta and the Lagrangian or green strain tensor is defined as

$$E_{ij} = \frac{1}{2} (\delta_{mn} \frac{\partial x_m}{\partial X_i} \frac{\partial x_n}{\partial X_j} - \delta_{ij}) \tag{3.4}$$

It is convenient to introduce displacement **u** defined by

$$u = x - X \tag{3.5}$$

Substitution of this into (3.4) gives the components of the Green strain tensor on the following form

$$E_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial X_j} + \frac{\partial u_j}{\partial X_i} + \frac{\partial u_m}{\partial X_i}\frac{\partial u_m}{\partial X_j}\right) \tag{3.6}$$

3.2 The Total Lagrangian Description

In the total Lagrangian description, the strains in two consequentive configurations C_n and C_{n+1} and all other configurations are related to the initial configuration C_o Figure 4 through the Green strain tensor E_o^n. The indices in E_o^n means configuration C_n refer to configuration C_o. Increments in strains from configuration C_n to C_{n+1} are given as the difference between Green strains in configurations C_n and C_{n+1}

$$\Delta E_o^{n+1} = E_o^{n+1} - E_o^n \tag{3.7}$$

Using (3.6) the above yields the Cartesian components of the Green strain increment

$$\Delta E_{ij} = \frac{1}{2}(\Delta u_{i,j} + \Delta u_{j,i} + u_{m,i}\Delta u_{m,j} + \Delta u_{m,i}u_{m,j} + \Delta u_{m,i}\Delta u_{m,j}) \tag{3.8}$$

Comma denotes differentiation with respect to the coordinate indicated by the following index. The variation of the total Green strains and the incremental Green strains are

$$\delta E_{ij} = \frac{1}{2}(\delta u_{i,j} + \delta u_{j,i} + u_{m,i}\delta u_{m,j} + \delta u_{m,i}u_{m,j}) \tag{3.9}$$

$$\delta \Delta E_{ij} = \frac{1}{2}(\delta u_{m,i}\Delta u_{m,j} + \Delta u_{m,i}\delta u_{m,j}) \tag{3.10}$$

The incremental stress-strain relation in total lagrangian description is given by

$$\begin{aligned} \Delta S_o^{n+1} &= C_T : \Delta E_o^{n+1} \\ S_o^{n+1} &= S_o^n + \Delta S_o^{n+1} \end{aligned} \tag{3.11}$$

Where C_T is the tangential or incremental constitutive tensor. S is commonly refered to as the second Piola-Kirchhoff stress tensor.

3.3 General Derivation of Finite Element Equilibrium Equation

3.3.1 Principle of Virtual Work

The equilibrium equation for an infinitesimal element (Figure 5.) may be integrated and expressed in terms of a virtual work equation for a finite body. The virtual work equation in a total Lagrangian formulation may be written

$$\int_{V_o} \mathbf{S} : \delta \mathbf{E} dV_o = \int_{\partial V_o} \mathbf{t}_o . \delta \mathbf{u} dA_o + \int_{V_o} \mathbf{f}_o . \delta \mathbf{u} dV_o \tag{3.12}$$

"δ" indicates "virtual" quantities or "variations". V_o and ∂V_o express the volume and the surface of the initial reference configuration. t_o and f_o are the prescribed surface traction and body forces per unit surface and volume of the initial configuration.

The virtual work equation for use in update Lagrangian description is expressed in terms of Cauchy stresses and strains in deformed configuration

$$\int_V \sigma : \delta \varepsilon dV = \int_{\partial V} \mathbf{t} . \delta \mathbf{u} dA + \int_V \mathbf{f} . \delta \mathbf{u} dV \tag{3.13}$$

V and ∂V express the volume and surface of the deformed configuration.

Consider the equilibrium of a general infinitesimal element such as in Figure 5. D'Alembert's principle may be combined with the virtual work expression (eqn. 3.12) in order to obtain the governing equations in dynamic applications.

$$\int_{V_o} \mathbf{S} : \delta \mathbf{E} dV_o + \int_{V_o} \rho_o \frac{d^2}{dt^2} \mathbf{u} \cdot \delta \mathbf{u}\, dV_o + \int_{V_o} C \frac{d}{dt} \mathbf{u} \cdot \delta \mathbf{u}\, dV_o = \int_D \mathbf{t}_o \cdot \delta \mathbf{u} d\Lambda_o + \int_{V_o} \mathbf{f}_o \cdot \delta \mathbf{u} dV_o \tag{3.14}$$

where the V_o and D express the volume and surface of initial reference configuration. **S** and **E** denote the stress and strain tensor. **u** and ρ is the displacement vector and mass density, where C is a matrix of secant damping coefficients. The body forces $\mathbf{f}_o$ and surface traction force $\mathbf{t}_o$ may be time dependent. Equation (3.14) also can be expressed as

$$R_{\text{int}} + R^I + R^D = R^E \tag{3.15}$$

where R_{int} is internal reaction force, R^I is inertia force, R^D is damping force and R^E is external force.

4. Nonlinear Dynamic Algorithm

4.1 FENRIS - General Purpose Nonlinear Finite Element System

FENRIS is a large scale, general program for nonlinear finite element analysis (ref. 2, 13 and 47). The program name is an abbreviation for Finite Element NonlineaR Integrated System. The development of FENRIS started in 1980 as a project between the Norwegian Institute of Technology (NTH), The Society for Industrial and Technical Research (SINTEF) and The Norwegian VERITAS. The main intention of the project has been to develop a new type of "truly nonlinear" program package. The program, as well as the technique of data storage, has been designed to correspond with the theoretical basis and major computational steps of nonlinear analyses. Another design concept for FENRIS has been to construct the program as a highly modular building kit. This has been done through isolating the various computational steps as much as possible from each other in the program software. This building block philosophy makes it easy to extend the program step by step and to readily accommodate new programming staff during development.

4.2 Nonlinear Dynamics Equilibrium Formulation

The element forces and loads are computed by use of (3.12) ,transformed to the global reference system and assembled for all elements. The dynamic equilibrium balance equations for time step i can be written as (ref. 10, 13)

$$R_{int,i} + R_i^I + R_i^D = R_i^E \tag{4.1}$$

The subscript int, supercript I, D and E denote internal forces, inertia force, damping forces and external forces. The internal reaction force vectors are computed element by element then transformed from the local co-rotated element coordinate system (ref. 13) to the global reference system and assembled for all elements. The inertia force vector and the damping force vector are also in reality computed element-wise and assembled with use of the global matrices M and C. The incremental form of the governing equation may be written as (ref.2, 13)

$$M_i\Delta\ddot{u} + C_i\Delta\dot{u} + K_{I,i}\Delta u = R_i^E - (R_{i-1}^I + R_{i-1}^D + R_{int,i-1}) \tag{4.2}$$

M_i is the mass matrix. C_i the incremental damping matrix and $K_{I,i}$ the incremental stiffness matrix. $K_{I,i}$ may be taken as the tangential element stiffness K_T,assembled to a global system matrix, or, the K_I may be used for several consecutive steps. $\Delta u_i, \Delta\dot{u}_i, \Delta\ddot{u}_i$ are the incremental displacement, velocity, and acceleration vectors. The Δ denotes finite but "small" increments corresponding to the difference between the two states considered. Equation (4.2) is used in connection with load increments and equilibrium iterations.

The program assumes that the Rayleigh damping matrix (ref. 14) is constructed from the following form

$$C = \alpha_1 M + \alpha_2 K_T + C_V \tag{4.3}$$

where α_1, α_2 are damping factors. and C_v is a diagonal matrix representing viscous dashpots prescribed at nodes.

4.3 Implicit Time Integration Algorithm

A general incremental-iterative scheme combined with the Newmark family of implicit time integration operators is used to solve the dynamic equations. The basic assumptions of the Newmark operators are (ref. 2, 15)

$$\dot{u}_{i+1} = \dot{u}_i + (1 - \gamma)\ddot{u}_i h + \gamma \ddot{u}_{i+1} h \tag{4.4}$$

$$u_{i+1} = u_i + \dot{u}_i h + (\frac{1}{2} - \beta)\ddot{u}_i h^2 + \beta \ddot{u}_{i+1} h^2 \tag{4.5}$$

Here h is the incremental time step

$$h = \Delta t = t_{i+1} - t_i$$

which may vary throughout the analysis. From equations (4.4) and (4.5) the corresponding increments of velocities and accelerations are

$$\Delta \dot{u}_{i+1} = \frac{\gamma}{\beta h} \Delta u_{i+1} - \frac{\gamma}{\beta} \dot{u}_i - (\frac{\gamma}{2\beta} - 1) h \ddot{u}_i \tag{4.6}$$

$$\Delta \ddot{u}_{i+1} = \frac{1}{\beta h^2} \Delta u_{i+1} - \frac{1}{\beta h} \dot{u}_i - \frac{1}{2\beta} \ddot{u}_i \tag{4.7}$$

Substitution equations (4.6) and (4.7) into equation (4.2) yields

$$\begin{aligned} \left[M \frac{1}{\beta h^2} + C \frac{\gamma}{\beta h} + K \right] \Delta u_{i+1} = {} & R^E_{i+1} - (R^I_i + R^D_i + R_{int,i}) \\ & + M\left[\frac{1}{\beta h} \dot{u}_i + \frac{1}{2\beta} \ddot{u}_i \right] + \mathbf{C}\left[\frac{\gamma}{\beta} \dot{u}_i + (\frac{\gamma}{2\beta} - 1) h \ddot{u}_i \right] \end{aligned} \tag{4.8}$$

By combining equations (4.3) and (4.8) the incremental - corrective equations become

$$\hat{K}\Delta\hat{u}_{i+1} = \Delta\hat{R}_{i+1} \tag{4.9}$$

where the effective stiffness matrix $\hat{K}$ is

$$\hat{K} = a_o M + c_o C_V + K_T \tag{4.10}$$

with the following constants

$$a_o = \frac{1}{1 + \frac{\alpha_2 \gamma}{\beta h}} \left(\frac{1}{\beta h^2} + \frac{\alpha_1 \gamma}{\beta h} \right) \tag{4.11}$$

$$c_o = \frac{1}{1 + \frac{\alpha_2 \gamma}{\beta h}} \left(\frac{\gamma}{\beta h} \right)$$

Here α_1 and α_2 are proportionality factors for the Rayleigh damping matrix (see equation 4.3). Note that the effective stiffness depends on the time step h. The increment of the effective load is

$$\begin{aligned} \Delta\hat{R}_{i+1} = R^E_{i+1} - R_{int,i} + M[\hat{a}_i + \hat{b}_i(\alpha_1 - a_o\alpha_2) - \ddot{u}_i] \\ + C_V\hat{b}_i(1 - c_o\alpha_2) \end{aligned} \tag{4.12}$$

where

$$\hat{b}_i = \left(\frac{\gamma}{\beta} - 1 \right)\dot{u}_i + \left(\frac{\gamma}{2\beta} - 1 \right)h\ddot{u}_i \tag{4.13}$$

$$\hat{a}_i = \frac{1}{\beta h}\dot{u}_i + \frac{1}{2\beta}\ddot{u}_i \tag{4.14}$$

The set of equations is solved with respect to the effective displacement increment $\Delta\hat{u}_{i+1}$. The increment of the real displacement vector is the

$$\Delta u_{i+1} = \frac{1}{1 + \frac{\alpha_2\gamma}{\beta h}}(\Delta\hat{u}_{i+1} + \alpha_2\hat{b}_i) \tag{4.15}$$

and $\Delta\dot{u}_{i+1}, \Delta\ddot{u}_{i+1}$ can be obtained from equation (4.6), (4.7). The corresponding total vectors are

$$\begin{aligned} u_{i+1} &= u_i + \Delta u_{i+1} \\ \dot{u}_{i+1} &= \dot{u}_i + \Delta\dot{u}_{i+1} \\ \ddot{u}_{i+1} &= \ddot{u}_i + \Delta\ddot{u}_{i+1} \end{aligned} \tag{4.16}$$

4.4 Iterative Schemes

If the vectors $u_{i+1}, \dot{u}_{i+1}$ and $\ddot{u}_{i+1}$ (eqn. 4.16) are substituted into the dynamic equilibrium equation (4.1), it will be found for nonlinear cases that this equation is not completely satisfied, and equilibrium iterations must be carried out. Figure 6 show the solution flow charts of FENRIS for the Nonlinear Transient Dynamic Analysis. The major solution procedures include element matrix generation/assembly, effective stiffness matrix ($\hat{K}$) triangulization and nonlinear load vector ($\hat{R}$) calculation. The matrix generation and the nonlinear load vector calculation tasks involve many decoupled calculations that are clearly well suited for parallel computations. The equation solver

adopted by FENRIS is the Cholesky decomposition, and there are four iterations strategies which can be used (see Figure 7). These are:

(1) **Incrementation with unbalanced force correction:** no iteration will be performed within each incremental step and the effective stiffness matrix $\hat{K}$ will be decomposed at every steps.

(2) **Initial stiffness iteration:** $\hat{K}$ matrix will not be updated and only decomposed once with the other steps using the backward/forward substitutions, this method is not well suited for parallel computations since the backward/forward substitutions is not efficiently carried out in parallel.

(3) **Modified Newton-Raphson:** $\hat{K}$ matrix will be updated and decomposed at each new incremental step.

(4) **True Newton-Raphson:** $\hat{K}$ matrix will be updated and decomposed at every iterations.

Two of these four methods, (1) Incrementation with unbalanced force correction and (4) true Newton Raphson show better potential for parallel implementations, however, the first method is not suitable for highly nonlinear problems.

4.5 CPU and Storage

The major time-consuming steps of implicit method are (1) nonlinear force vectors generation, (2) input/output, and (3) matrix decomposition. The mass matrix, stiffness matrix, viscous damping matrix, internal force vectors can be computed element by element and assembled into the global matrices M, K, C_V and R_{int}. These computational tasks are independent and highly parallelizable and can be assigned across processors. The Cholesky decomposition step is the most time-consuming calculation for large size

finite element problems, and this method has shown good speedups through parallel and vector processing, especially when the half-bandwidth is sufficiently large (ref. 22, 40).

For a large structural systems the effective stiffness matrix cannot be accommodated in core storage (shared memory), and must be segmented into blocks (ref. 43). These blocks must then be stored temporary on low speed auxiliary storage, usually as disk files, where the CPU time is relatively high. Shared/Local memory can take advantage of the local memory available on each processor, Data blocks are stored in local core memory by use of standard FORTRAN 77 calculations such as Local Variables 1 to n equal to Share Variables 1 to n, these data mapping CPU times are much less than the standard I/O operation. Therefore, the use of local memory as a secondary storage unit is needed for parallel finite element implementations. More details of this concept will be discussed in the next chapter.

The equation (4.9) may take the form as A X = B, where the effective stiffness matrix $A = L\,U = L\,D^{-1}\,L^{T}$. Here the matrix L is a lower unit triangular matrix and D is a diagonal matrix, the decomposition is carried out in three step:

$$
\begin{aligned}
l_{ij} &= a_{ij} - \sum_{n=1}^{i-1} l_{ni}\, l_{nj} \qquad i < j \\
l_{ij} &= l_{ij}\; d_{ii}^{-1} \\
d_{jj}^{-1} &= 1 \,/\, (a_{jj} - \sum_{n=1}^{j-1} l_{nj}\, l_{nj}\, d_{nn}) \qquad \text{for all j}
\end{aligned}
\tag{4.17}
$$

The matrix L has the same shape as A (skyline form) and is therefore stored in the same location. The diagonal matrix D^{-1} is stored in an array residing in core. The stiffness matrix A is divided into several blocks, each consisting of several columns and A is

triangulized block by block. Some typical results of this algorithm are shown on the following section.

5. Application to FEM Analyses

5.1 Sequential Verification

The FENRIS system has been initially installed at Georgia Tech on VAX/750 VMS and FLEX/32 UNIX operating systems; and initial results for the S-Frame test problem have been obtained on both machines. The test finite element model comprises 180 triangular shell elements with 456 unknowns. The impact phenomenon was computed on the basis of elastic-perfectly plastic constitutive material (A-36 Steel). Dynamic computations were performed via a matrix solution of Newmark's unconditional stable scheme, with a time increment equaled to 0.0005 second. A fine mesh or smaller time increment was not used because the limitation of CPU time and disk space on the VAX/750 and FLEX/32. Typical results are shown as Figure 8, 9; these results compare with favorably with similar results by Argyris, et al, (ref. 46) and indicate that FENRIS system provides an adequate basis for the parallel crash dynamics investigation.

5.2 Static Analysis

The FENRIS parallel version one (FENRIS/P1, Figure 10) with a parallel $LD^{-1}L^{T}$ (Figure 11) has been developed and installed on FLEX/32 MMOS (Multitasking Multicomputing Operating System) at Georgia Tech CAD/CAE Laboratory. Speedups of this $LD^{-1}L^{T}$ decomposition (compare with the sequential FENRIS $LD^{-1}L^{T}$ decomposition) are very encouraging. The improvements are achieved by refined parallel computation strategies, and a highly machine independent waiting control routines were used which reduce waiting time in the matrix decomposition. A sample static test problem is shown at Figure 12. which is a cantilever beam subjected to a concentrated force. This finite element model comprises 100 quadrilateral elements with 444 unknowns and the

maximum half-bandwidth equal to 48. Typical results showing the improved computation speedups are given in Figure 13. The results show computation speedups of up to 6.54 for seven processors.

5.3 Crash Dynamic Analysis

A parallel finite element approach to the crash test problem of Figure 2 was also carried out which is an automobile front end S-Shape torque box beam subjected to impact. This finite element model comprises 180 triangular shell elements with drilling freedoms and amounting to 456 degrees-of-freedom, the maximum half-bandwidth equal to 42. Typical results showing the improved computation speedups are given in Figure 14. The results show computation speedups of up to 6.32 for seven processors and indicate that the significant speedups can be achieved through the use of many processors for an appropriate finite element problem. It should be noted that the Cholesky decomposition is also well-suited for vector computation (Figure 15). In the parallel Cholesky decomposition, each processor is responsible for decomposing the whole column of the system matrix; therefore no vector length penalty is introduced. From the above observations the $LD^{-1}L^{T}$ algorithm shows a good promise for parallel/vector computation and indicate that a Cholesky based finite element system such as FENRIS appears well suited for parallel/vector implementations.

The computation times attributed to both I/O and CPU for the S-Frame test problem are shown at Figure 16. This Figure illustrates that the percentage of computation associated with effective stiffness matrix, internal and external force vectors generation is 84.6 % whereas for equation solving is 11.90% Most of the times are spent on effective stiffness matrix, nonlinear force vectors generation and I/O. These parts will

be discussed at next chapters and it is believed that high parallel efficiency can be achieved.

6. The Parallel Element Generation Theory

6.1 Element Formulation

The main objective is now to come up with a formulation of the element stiffness generation, the virtual work principle for dynamic problems is shown at equation (3.12). The incremental form of the virtual work equation leads to an incremental equilibrium equation for the element. The strain, stress tensor, the surface traction force, body force vector at the incremental n+1 can be expressed as

$$\begin{aligned} E^{n+1} &= E^n + \Delta E \\ S^{n+1} &= S^n + \Delta S \\ t^{n+1} &= t^n + \Delta t \\ f^{n+1} &= f^n + \Delta f \end{aligned} \tag{6.1}$$

the inertia force and damping force vector at incremental step n+1 can be written as

$$\begin{aligned} R^I_{n+1} &= R^I_n + \Delta R^I \\ R^D_{n+1} &= R^D_n + \Delta R^D \end{aligned} \tag{6.2}$$

where

$$\begin{aligned} R^I &= \int_{V_o} \rho_o \frac{d^2}{dt^2}\, \mathbf{u} \,.\, \delta\mathbf{u}\, dV_o \\ R^D &= \int_{V_o} C \frac{d}{dt}\, \mathbf{u}.\, \delta\mathbf{u}\, dV_o \end{aligned} \tag{6.3}$$

Most solution procedure for nonlinear problems employ a linearized incremental form of the equilibrium equations, put equation (6.1), (6.2) and (6.3) into (3.12) gives the incremental virtual work equation for dynamic problems in using Total Lagrangian description:

$$\int_{V_o}(\mathbf{S}^n:\delta\Delta\mathbf{E} + \Delta\mathbf{S} : \delta^n\mathbf{E} + \Delta\mathbf{S} : \delta\Delta\mathbf{E})dV_o + \Delta R^I + \Delta R^D = \int_{A_o}\Delta\mathbf{t}_o.\delta\mathbf{u}dA_o + \int_{V_o}\Delta\mathbf{f}_o.\delta\mathbf{u}dV_o \tag{6.4}$$

The desired linear form of (6.4) is obtained by neglecting the quadratic Δ term, hence

$$\int_{V_o}(\mathbf{S}^n:\delta\Delta\mathbf{E} + \Delta\mathbf{S} : \delta^n\mathbf{E})dV_o + \Delta R^I + \Delta R^D = \int_{A_o}\Delta\mathbf{t}_o.\delta\mathbf{u}dA_o + \int_{V_o}\Delta\mathbf{f}_o.\delta\mathbf{u}dV_o \tag{6.5}$$

The corresponding equation in Updated Lagrangian description is

$$\int_{V_n}(\sigma^n:\delta\Delta\mathbf{E} + \Delta\mathbf{S} : \delta^n\varepsilon)dV + \Delta R^I + \Delta R^D = \int_{A_n}\Delta\mathbf{t}.\delta\mathbf{u}dA + \int_{V_n}\Delta\mathbf{f}.\delta\mathbf{u}dV \tag{6.6}$$

The first in equation (6.5) and (6.6) depends on the current state of stress and gives rise of geometry nonlinear stiffness matrix. The second term is the basis for the incremental material nonlinear stiffness matrix and depends on the incremental material law. The tangential or incremental element stiffness matrix is now found to be

$$
\begin{aligned}
\delta \mathbf{v}^T \mathbf{K}_7 \delta \mathbf{v} &= \int_{V_0} (\mathbf{S}^n : \delta \Delta \mathbf{E} + \Delta \mathbf{S} : \delta^n \mathbf{E}) dV \\
\delta \mathbf{v}^T \mathbf{K}_{\text{Geom}} \delta \mathbf{v} &= \int_{V_0} (\mathbf{S}^n : \delta \Delta \mathbf{E}) dV \\
\delta \mathbf{v}^T \mathbf{K}_{\text{mat}} \delta \mathbf{v} &= \int_{V_0} (\Delta \mathbf{S} : \delta^n \mathbf{E}) dV \\
K_T &= K_{\text{mat}} + K_{\text{Geom}}
\end{aligned}
\tag{6.7}
$$

$$
(K_{\text{elem}})_{\text{Global}} = T^T (K_{\text{elem}})_{\text{Local}} T \tag{6.8}
$$

$$
K_{\text{System}} = \sum_{\text{elem}=1}^{n} (K_{\text{elem}})_{\text{Global}} \tag{6.9}
$$

T is the transformation tensor.

6.2 Parallel Element Matrix Generation

Element generation in finite element analysis is carried out at the element level and then assembled to produce the system stiffness matrix (eqn. 6.7 - 6.9). These calculations can be, in principle, carried out in parallel without any synchronization. Practically, after the elemental calculations have been achieved, assembling these element stiffness matrices into the system stiffness matrix syncronization is required along a shared boundary. In Figure 17, a mesh node b is shown attached to four distinct elements E_1, E_2, E_3, E_4 and each element belonging to a different processor. From equations 6.9 one can easily find that syncronization points appear on the node a, b, c, d and e, especially on node b. A mesh coloring method has been proposed by Farhat and Crivelli (ref. 18), to minimize the syncronization delay; however, an efficient processor mapping and an efficient FEM mesh modeling will be required by their method. Figure 18, 19 (ref. 18) show two different mesh model for the same physical problem; To obtain a good parallel performance and accurate results, one needs a comprehensive knowledge of parallel computations and finite element methods (the best parallel mesh model does not

necessarily give the best finite element mesh model). These make the mesh coloring method less practical for general purpose parallel finite element system. An easy way to avoid above restrictions and minimize the syncronization delay is: first calculate all the element stiffness matrix (eqn. 6.7 and 6.8) and put into a memory buffer, second perform the equation 6.9 to assemble all the element stiffness matrix. Several useful methods will be discussed on the following sections.

6.3 Hybird Parallelism

In this method the element by element (EBE) parallelism will be applied to equation (6.7) and (6.8) for element stiffness generations, and the substructure parallelism will be used in equation (6.9) for system matrix assembly. Figure 20 shows two typical processor mapping for a cantilever beam subjected to a concentration load. Since the calculation works on each processor are independent therefore processor mapping is not important for this parallel procedure. Each processor calculates its own element one by one and puts it into memory buffer which is shown at Figure 21. After the shared memory buffer is filled these data will be mapped into local memory buffer. This shared-local data mapping concept (Figure 22) has been applied to the FENRIS and implemented on FLEX/32 shared memory multicomputers. The results shown that several order of magnitude speedup have been achieved (compare with standard disk I/O), however, one thing should be noted that this speedup performance is machine dependent and requires local memory (or local registration address) available; otherwise standard disk I/O should be applied. The Assembly of the element stiffness matrices into the system stiffness matrix is performed by a stand alone subroutine, in this assemble procedure algorithm will require several necessary control data from preprocessing, usually theses data (such as D.O.F per node and per element, No. of integration points and K matrix starting pointer etc.) will be store in core or in a certain file (for a large

finite element model). An efficient way to assemble system stiffness matrix is go by subdomain parallelism. Focus syncronization points on the subdomain shared boundary will more easier than trace these points element by element, since the interaction boundary is shared by more than one processor therefore the parallel control should be applied in order to obtain correct results. More details of this parallel merge algorithm will be discussed on the next two section.

6.4 Subdomain Parallelism

An alternative way to generate element stiffness matrices and to assemble the system stiffness matrix is through subdomain parallelism. Figure 23 shown a typical processor mapping for this algorithm. Here each processor is responsible for a certain subdomain and calculates all element stiffness matrices and assembles them into sub-system stiffness matrices. All nodes that belong to local memory except the boundary nodes (such as $B.N_1, B.N_2, B.N_3, B.N_4$) are put into a shared memory buffer to obtain a complete sub-system stiffness matrix. Figure 24 shown the data structure of the Shared-Local memory arrangement. the system stiffness matrix is usually stored in shared memory for matrix decomposition so the local stiffness matrix needs to be mapped back to the shared memory. For the case of shared memory parallel computers that do not have local memory available, the subdomain parallelism will need make some modification so that each processor generates its own K_{elem} and K_{sub} both in the shared memory. In actual fact the element by element stiffness generation method is a special case of the second subdomain parallelism approach and these two algorithm can be merged to one.

6.5 Merge of System Stiffness Matrix

Figure 25 shown a cantilever beam split by four equal subdomains, in this model

each element has four nodes and each node has 6 D.O.F with nodes 1 to 4 fixed. The overlap boundary nodes between two processors are nodes 17 to 20, 33 to 36, 49 to 52 and the overlap D.O.F are 73 to 96, 169 to 192, 265 to 288. Figure 26 shown the K_{system} matrix assemble procedure. Each processor assemble its own sub-system stiffness matrix and put a parallel control such as LOCK-ON and LOCK-OFF along the boundary nodes. When a boundary node is calculating, the flag will be set on LOCK-ON mode at that particular node, that is mean no other processor can access the memory address of that particular node until the flag is set at LOCK-OFF. Figure 27 shown a irregular subdomain where the boundary nodes are not continueous and more book keeping effort is required. A preprocessing automesh decomposition has been proposed by Malone (ref. 19), and this algorithm will be the baseline automesh decomposition of later version of Parallel FENRIS.

6.6 Concluding Remarks

Two algorithms for parallel element stiffness generation have been presented and discussed, the subdomain parallelism (which uses local memory) and the element by element parallelism. The first algorithm has advantages for the parallel/vector computations where each processor can group a set of elements (such as 64 or 128 elements) and then perform the vectorization procedure. The second algorithm appears less suited for parallel/vector computations unless a very large shared memory available, e.g. for a 64 processors parallel computer. If each processor group of 128 elements then the shared memory will be required large enough to accommodate 8192 elements (equal to 2,654,208 double precision variables or 21,233,664 bytes for a group of three nodes triangular shell element with 6 D.O.F per node where the $K_{element}$ matrix contains 18 x 18 variables) in core at the same time. However, this is not an issue for high-end supercomputer such as CRAY Y-MP, CRAY-3 and IBM 3090-600E, and it is believed that

a huge in core memory mid-range parallel supercomputer will be available very soon. The results show that both approaches for parallel/vector generation of element matrices are feasible for realistic shared/local memory computers.

7. Internal Force Vector Generation

Force vectors generations are included traction forces, body forces and internal forces generations etc. The major time consuming of these force vectors generations is the generation of internal force. The internal nodal reaction forces for an element, S_{int} are found from the left hand side (first term) of the virtual work expression. Considering e.g. the TL-description (Total Lagrangian) Equation (3.12) yields

$$\mathbf{S}_{\text{int}} = \int_{V_o} \mathbf{B}^{\mathrm{T}} \mathbf{S} dV_o \tag{7.1}$$

where B is the strain giving matrix for an element.

The above equation is also valid for the Co-rotated Lagrangian description. Similarly, S_{int} for the UL-description (Update Lagrangian), is found by use of equation (4.6)

$$\mathbf{S}_{\text{int}} = \int_{V} \mathbf{B}^{\mathrm{T}} \sigma dV \tag{7.2}$$

The reaction forces are thus obtained simply by integration of the internal state of stress, and also can be expressed as the following formulation

$$\mathbf{R}_{\text{int}} = \int_{\substack{-1\le\xi\le 1 \\ -1\le\eta\le 1 \\ -1\le\zeta\le 1}} \mathbf{B}^{\mathrm{T}} \mathbf{S} \det \mathbf{J} d\xi d\eta d\zeta \tag{7.3}$$

The internal force vectors are computed element by element then transformed from the local co-rotated element coordinate system to the global reference system and assemble for all elements. Figure 28 shown the subdomain parallelism and Figure 29 shown the element by element parallelism for computing internal force vectors. Practically, element stiffness generation and internal force vectors generation will use the same parallel algorithm.

7.1 Concluding Remarks

Figure 30 shown a cantilever plate subjected by a concentrated load. This model was performed by FENRIS quadrilateral shell element and strain hardening material library. The first load is within the elastic zone, the second load is located at the plastic zone, there is slightly different CPU times between these two loads (for some material library and element type the difference of CPU times is larger than this model). For a fine parallel algorithm especially applied to massively parallel computers the load balance is very important. For the S-Frame crash test problem (ref. 10), after impacted some element within the elastic zone and some element within the plastic zone, it is not easy to control the work load balance by using of subdomain parallelism, in this case element by element parallel algorithm shown a better feature than the subdomain parallelism. In summary chapter 4 and 5, element by element parallelism will be used in the Parallel FENRIS version II (FENRIS/p2).

8. Summary

This report summarizes an investigation and evaluation several potential algorithms for Parallel FENRIS which appear well suited for stiffness matrix generation and internal force vectors generations on shared memory parallel supercomputers. The main concerns of this research are to investigate the potential of parallel computers to significantly improve the capability for finite element crash analysis of automotive vehicles. Task planned are:

(1) the development of concurrent algorithms for nonlinear transient analysis which make effective use of multiprocessor computing power. This algorithms include (a) implicit methods such as Newmark type which build on Cholesky decomposition strategy and exhibits "algorithmic parallelism", and (2) explicit method such as the central difference time integration scheme which exhibits "physical parallelism"

---------- **Done 1985-1986 and Reference 21.**

(2) An assessment of the potential of the shared memory and local memory computers for use in parallel computations

---------- **Done 1986 and Reference 22, 23.**

(3) Implement the FENRIS nonlinear finite element commercial program on the FLEX/32 MMOS (Multicomputing Multitasking Operating System) for parallel solution

---------- **Done 1986-1987 and Reference 10.**

(4) Incorporate a parallel Cholesky decomposition method in FENRIS and evaluate the performance of the resulting parallel nonlinear finite element system and evaluation this method with the GM S-Frame test problem

---------- **Done 1987-1988 and Reference 10, 11 and 20.**

(5) Investigate the implementation and performance of a parallel force vectors generation in FENRIS

---------- **This Report**

In summary, work to date has progressed well on several fronts toward the research objectives. A baseline software system, good parallel crash dynamics methods, and validation results on the test problem have all achieved. The initial results are encouraging and indicate that the key computation steps in crash analysis can benefit significantly from parallel computers. More detailed studies should be carried out on realistic problems with all major computation step being implemented in parallel.

References

1. Noor, A., Storaasli, O., and Fulton, R. Finite Element Technology in the Future. Impact of New Computations on Computational Mechanics. (Noor and Pilkey, Editors), ASME Special Publication H00275 (November 1983) pp. 1 - 32. .of

2. FENRIS (Finite Element NonlineaR Integrated System) System Manual, Theory - Program outline - Data Input, by NTH-SINTEF-VERITEC, Norway, 1987.

3. MSC/NASTRAN Application Manual, McNeal-Schwendler Corp., Los Angeles, CA.

4. ANSYS Theoretical Manual, Swanson Analysis System Inc. 3rd Edition, March 1, 1986.

5. Komzsik, L., Parallel Decomposition Technique on a Double level Storage System, 1986, International Conference for Vector and Parallel Processing, Loen, Norway.

6. George, A., Heath, M.T., and Liu, J., Parallel Cholesky Factorization on a Shared-Memory Multiprocessor. Tech Report ORNL-6124, Oak Ridge National Laboratory, March 1985.

7. Farhat, C., and Wilson, E., A Parallel Active Column Equation Solver, Computers and Structures Vol. 28. No. 2, pp. 289-304, 1988

8. Heath, M. T., Parallel Chelosky Factorization in Message-Passing Multiprocessor Enviroments, Tech. Rept. ORNL-6150, March 1985, Oak Ridge National Laboratory.

9. Duff, I. S., Parallel Implementation of Multifrontal schemes, Parallel computing Vol. 3. pp 193-204, 1986.

10. Fulton, R. E., and Chiang, K. N., Computational Crash Dynamics Methods for Fifth Generation Supercomputers. GM Project E-25-633 Final Reports Phase II, Jan. 1988.

11. Fulton, R. E., Chiang, K. N., and Goehlich D., Parallel Computer Implementation of Finite Element Methods, 2nd International Conference on Vector and Parallel Computing Issues in Applied Research and Development, Tromso, Norway, June 6-10, 1988.

12. Mollestad, E., Techniques for Static and Dynamic Solution of Nonlinear Finite Element Problems, The Norwegian Institute of Technology, Division of Structural Mechanics, Report No. 83-1 December 1983.

13. Nygard, M. K., The Free Formulation for Nonlinear Finite Elements with Applications to Shells, Report No. 86-2, Division of Structural Mechanics, The Norwegian Institute of Technology, Norway, Dec. 1986.

14. Clough, R. W. and Penzien, J., Dynamics of Structures, McGraw-Hill, 1975.

15. Newmark, N. A Method of Computation for Structural Dynamics. J. Eng. Mech. Div., ASCE (July 1959) EM3, pp. 67-94.

16. Fung, Y. C., Foundations of Solid Mechanics, Prentice-Hall, Inc.

17. Fulton, R. E., and Chiang, K. N., Computational Crash Dynamics Methods for Fifth Generation Supercomputers. GM Project E-25-633 Final Reports Phase II, Jan. 1988.

18. Farhat, C., and Crivelli, L., A General Approach to Nonlinear Finite Element Computations on Shared Memory Multiprocessors, Documentation CU-CSSC-87-09, Center for Space Structures and Controls, College of Engineering, University of Colorodo, 1987.

19. Malone, J. G., Automated Mesh Decomposition and Concurrent Finite Element Analysis for Hypercube Multiprocessor Computers, G.M. labs. Report, Engineering Mechanics Department, Warren, MI. May 18, 1987.

20. Goehlich D., Chiang K. N. and Fulton R. E., Parallel Computer Approach to Finite Element Methods, UPCAEDM 1988 Conference, pp. 139-144, June 27-29, 1988, Atlanta, Georgia.

21. Fulton, R. E., and Chiang, K. N., Computational Crash Dynamics Methods for Fifth Generation Supercomputers. GM Project E-25-633 Final Reports Phase I, Dec. 1986.

22. Chiang, K. N., and Fulton, R. E., Nonlinear Dynamics Methods for Parallel Computers. ASCE Fifth Conference on Computing in Civil Engineering, March 29-31, 1988, Alexandria VA.

23. Fulton, R. E., and Chiang, K. N., Comparison of Shared Memory and Hypercube Architectures for Structural Dynamics, Third International Conference on Supercomputing and Second World Supercomputer Exhibition, May 15-20, 1988, Boston, Massachusetts.

24. Fredriksson, B., and Mackerle, J. Partial List of Major Finite Element Programs and Description of Some of Their Capabilities. In State-of-the-Art Surveys on Finite Element Technology (Noor and Pilkey, Editors). ASME Publication H00290, (1983) pp. 363-403.

25. Storaasli, O., Peeples, S., Crockett, T., Knott, J., and Adams, L. The Finite Element Machine: An Experiement in Parallel Processing. NASA TM 84514 (1982).

26. Matelan, N., The FLEX/32 Multicomputing Enviroment. In Research in Structure and Dynamics--1984. NASA CP 2335 (October 1984) pp. 1 - 14.

27. Storaasli, O., Ransom, J., and Fulton, R. Structural Dynamic Analysis on a Parallel Computer: The Finite Element Machine. AIAA Paper 84- 0966-CP, presented at 25th AIAA/ASME/AHS Structures, Structural Dynamics and Materials Conference, Palm Springs, CA (14-16 May 1984)

28. Ransom, J., Storaasli, O., and Fulton, R. Application of Concurrent Processing to Structural Dynamic Response Computations. In Research in Structures and Dynamics--1984, NASA CP 2335 (Oct. 1984) pp. 31-44

29. Bostic, S., and Fulton, R. E., A Concurrent Processing Approach to Structural Vibration Analysis. 26th AIAA/ASME/ASCE/AHA Structures, Structural Dynamics and Materials Conference, Orlando, FL (15-17 April 1985).

30. Adams, L., and Ortega, J. A Multicolor SOR Method for Parallel Computation. Proc. 1982 Int. Conf. Parallel Processing, pp. 53-56.

31. Adams, L., and Jordon, H. Is SOR Color-Blind? ICASE Resport 84-14, NASA - Langley Research Center (1984).

32. Johnson, O., Micchelli, C., and Paul, G. Polynomial Preconditioners for Conjugate Gradient Calculations. SIAM J. Num. Anal. 20 (1983) pp. 362 - 376.

33. Adams, L. Iterative Algorithms for Large Sparse Linear System on Parallel Computers. Ph.D. Thesis, University of Virginia (1982).

34. Adams, L. An M-Step Preconditioned Conjugate Gradient Method for Parallel Computation. Proc. 1983 Int. Conf. Parallel Processing, pp. 36 - 43.

35. Adams, L. M-Step Preconditioned Conjugate Gradient Methods. To Appear in SIAM J. Sci. Stat. Comp.

36. Schreiber, R., and Tang, W. Vectorizing the Conjugate Gradient Method. Proc. Symposium Cyber 200 Applications, Ft. Collins, CO (1982)

37. Poole, E., and Ortega, J. Incomplete Cholesky Conjugate Gradient on the Cyber 203/205. Proc. 1984 Cyber 205 Applications Seminar.

38. Bostic, Susan W., and Fulton, R. Implementation of the Lanczos Method for Structural Vibration Analysis on a parallel Computer. AIAA Paper No. 86 - 0930 - CP, AIAA/ASME/ASCE/AHS 27th Structures, Structural Dynamics and Materials Conference, San Antonio, TX (19-21 May 1986).

39. Fulton, R. E. The Impact of Parallel Computing on Finite Element Computations. Presented at International Conference on Reliability of Methods for Engineering Analysis, Swansea, U.K. (9-11 July 1986).

40. Goehlich, D., Komzsik, L., and Fulton, R. E., Decomposition of Finite Element Matrices on Parallel Computers. To be presented at ASME Computers in Engineering Conference, August 10-14, 1987, N.Y.

41. Geist, A., Solving Finite Element Problems With Parallel Multifrontal Schemes, Second Conference on Hypercube Multiprocessors, September 29 - October 1, 1986, Knoxville, Tennessee.

42. Ni, Chi-Mou Impact Response of Curved Box Beam-Column with Large Global and Local Deformations. AIAA Paper No. 73 - 401, AIAA/ASME /SAE 14th Structures, Structural Dynamics, and Materials Conference, Williamsburg, Virginia/March 20-22, 1973.

43. Wilson, E. L., and Dovey, H. H., Solution or Reduction of Equilibrium Equations for Large Complex Structural Systems, Journal of Advances in Engineering Software, 1978 Vol. 1, No. 1 pp. 19-25.

44. Mondkar, D. P., and Powell G. H., Large Capacity Equation Solver for Structural Analysis, Journal of Computers & Structures, 1974. Vol. 4 pp. 699 - 728.

45. Lyzenga, G. A., Raefsky, A., and Hager, B. H., Finite Elements and the Conjugate Gradients on a Concurrent Processor, CalTech/JPL Report C3P-119, Dec. 1984.

46. Argyris, J., Balmer, H. A., Doltsinis, J. St., and Kurz, A., Computer Simulation of Crash Phenomena, International Journal of Numerical Methods in Engineering, Vol. 22, pp. 497-519, 1986.

47. Storaasli, O. O., and Bergan, P., A Nonlinear Substructure Method for Concurrent Processing Computers, AIAA Publ. 86-0852, 1986.

48. Bathe, K. J., Finite Element Procedures in Engineering Analysis, Prentice - Hall Inc., NJ, 1982.

49. Dhatt, G., and Touzot, G., The Finite Element Method Displayed, John Wiley & Sons Inc., NY. 1984.

50. Cook, R. D., Concepts and Applications of Finite Element Analysis, Second Edition, John Wiley & Sons Inc., NY, 1981.

51. Irons, B. M., A Frontal Solution Program for Finite Element Analysis, Int. J. Num. Meth. Engrg., Vol 2. pp. 5-32, 1970.

52. Ashcraft, C. C., Parallel Reduction Methods for the Solution of Banded Systems of Equations, Computer Science Dept., Research Publi. GMR - 5094, June 30, 1985., Warren, MI.

53. Ni, Chi-Mou, A General Purpose Analytical Technique and Program, 'NONDRIS', for Nonlinear Dynamic Response of Integrated Structures, E. M. Dept., Research Report EM-507, April 20, 1981., Warren, MI.

54. Intel iPSC System Overview Manual, Order No. 175278 - 001.

55. Abu-Shumays, I. K., Comparison of Methods and Algorithms for Tridiagonal Systems and for Vectorization of Diffusion Computations. Bettis Atomic Power Lab., DE-AC11-76PN00011, W. Mifflin, PA.

56. Kershaw, D. Solution of Single Tridiagonal Linear Systems and Vectorization of The ICCG Algorithm on the CRAY-1, Lawrrence Livermore National Lab., ISBN 0-12-592101-2, Livermore, CA. 1982.

57. Katona, M. G. and Zienkiewicz, O. C., A Unified Set of Single Step Algorithms Part 3: The Beta-m Method, A Generalization of the Newmark Scheme, Int. J. Num. Meth. Engrg. Vol 21. pp 1345-1359, 1985

58. D'Souza, A. F. and Garg, V. K., Advanced Dynamics Modeling and Analysis, Printice-Hall Inc., NJ, pp 198-228, 1984.

59. Duff, I. S. and Reid, J. K., The Multifrontal Solution of Indefinite Sparse Symmetric Linear Systems, ACM Trans. on Math. Software, Vol.9 pp. 302-325

60. Hilton, E. and Owen, D. R. J., Finite Element Programming, Academic Press. NY, 1979.

61. Bryson, J. W., ORVIRT.PC: A 2-D Finite Element Fracture Analysis Program for a Microcomputer,, ORNL-6208, Oct. 1985.

62. Brebbia, C. A., Finite Element System, A Computational Mechanics Publication, NY, 1985.

63. Parlett, B. N., Nour-Omid, B., Implementation of Lanczos Algorithms on vector computers, N00014-76-C-0013, UC Berkeley, CA, 1985.

64. Hallquist, J. O., Theoretical Manual for DYNA3D, Lawrence Livermore Lab., CA, March 1983.

65. O'Leary, P. O., Parallel Implementation of the Block Conjugate Gradient Algorithm, Journal of Parallel Computing, pp 127-139, 1987.

66. Henk A. van der Vorst, Large Tridiagonal and Block Tridiagonal Linear System on Vector and Parallel Computers, Journal of Parallel Computing, pp 45-54, 1987.

67. Delves, L. M. and Sanba, A. S., Band Matrices on the DAP, GR/B/24332, University of Liverpool and University of Birmingham.

68. Meurant, G., Multitasking the Conjugate Gradient Method on the CRAY X-MP/48, Parallel Computing, Vol. 5, pp 267-280, 1987.

69. Benson, D.J., Hallquist, J.O. and Stillman, D.W., DYNA3D, INGRID and TAURUS - An Integrated, Interactive Software System for Crashworthiness Engineering, Lawrence Livermore Lab. CA, W-7405-Eng-48.

70. Bostic, S. W. and Fulton R. E., Experience with the Lanczos Method on a Parallel Computer, ASME Computers in Engineering Conference, August 9-13, 1987, N.Y.

71. Parkinson, D., Organizational Aspects of Using Parallel Computers, Parallel Computing Vol. 5, pp 75-83, 1987.

72. Hockney, R. W., Parametrization of Computer Performance, Parallel Computing Vol, 5, pp 97-103, 1987.

73. "FLEX/32 Multicomputer - System Overview", Document 030-000-001, Flexible Computer Corporation, Dallas, Texas, 1985.

74. Hughes, T. J. R., The Finite Element Method Linear Static and Dynamic Finite Element Analysis, Printice-Hall International Editions, 1987.

75. Supercomputers: The Proliferation Begins, Electronics pp 51-77, March 3, 1988.

76. Quinn, M. J., Parallel Sorting Algorithms for Tighly Coupled Multiprocessors, Parallel Computing Vol. 6, pp 349-357, 1988.

	Maximum No. Processors	Vector	*Memory
High Speed Systems	** (2400 -		
Cray Y-MP	8 3600)	*	S
IBM 3090	6 (900)	*	S
ETA	10 (10,286)[e]	*	S/L
Massively Parallel			
BB&N GP1000	256 (125) **		L
INTEL iPSC/2 VX	64 (1028)	*	L
N Cube 10	1024(500)		L
Amatek 2010	512 (80)	*	L
Connection Machine	65,536 (2500)		L
Meiko Computing Surface	1000 and up (1000 or more)		Transputer
Intemediate Systems			
Alliant FX/80	8 (189) **	*	S
Elxsi 6420	10 (120)		S
Convex C240	4 (200)	*	S
FLEX/32	20 (80)	*	S/L
High Performance Workstations			* * *
Ardent TITAN	4 (64) **	*	S (128)
Stellar GS-10000	4 (40)	-	S (128)
Raster/Sun	8 (160)	*	L (128)
Apollo DN-10000	4 (144)	-	S (128)
Pixel	82 (820)	-	S (128)
Silicon Graphics	2 (40)	-	L (41)
S = Shared, L = Local	** MFLOPS	*** MBYTE Memory e - Estimated	

Table 1. Parallel Computer Family

IBM Becoming More Visible

- Chen Joints IBM to Lead New Initiative
- 3090 Being Marketed as Parallel
- IBM Parallel Fortran
- Bergen Scientific Center, Norway Conference
- Marketing Parallel Computing Credible

NASA Langley Developing a CRAY Type FE System

- Documented, Prototype Software
- Buying Cnvex/CRAY 2 System

Several Parallel Conferences

● Boston	Computer Science Focus
● Tromso, Norway	Lots of Hypercube Activities
● Tokyo	Graphics Workstation

Sandia Produces Massively Parallel Structures FEM Example of Plane Stress Cantilever Beam

- Demos 1024 Processor NCUBE Hypercube which Gives 500 Speedup Over 1 Processor Solution

Table 2. Other Parallel Computer Activities

Machine	Date	Number of Processors	Rate Speed MFLOPS	Memory MWORDS
CRAY-1	1976	1	160	1
X-MP/2	1982	2	420	4
X-MP/4	1984	4	840	8
CRAY-2	1985	4	1,700	128
Y-MP	1988	8	2,500	32
CRAY-3	1989	16	16,000	572
CRAY-4	1992	64	128,000	2048

Table 3. CRAY Supercomputer History/Plans

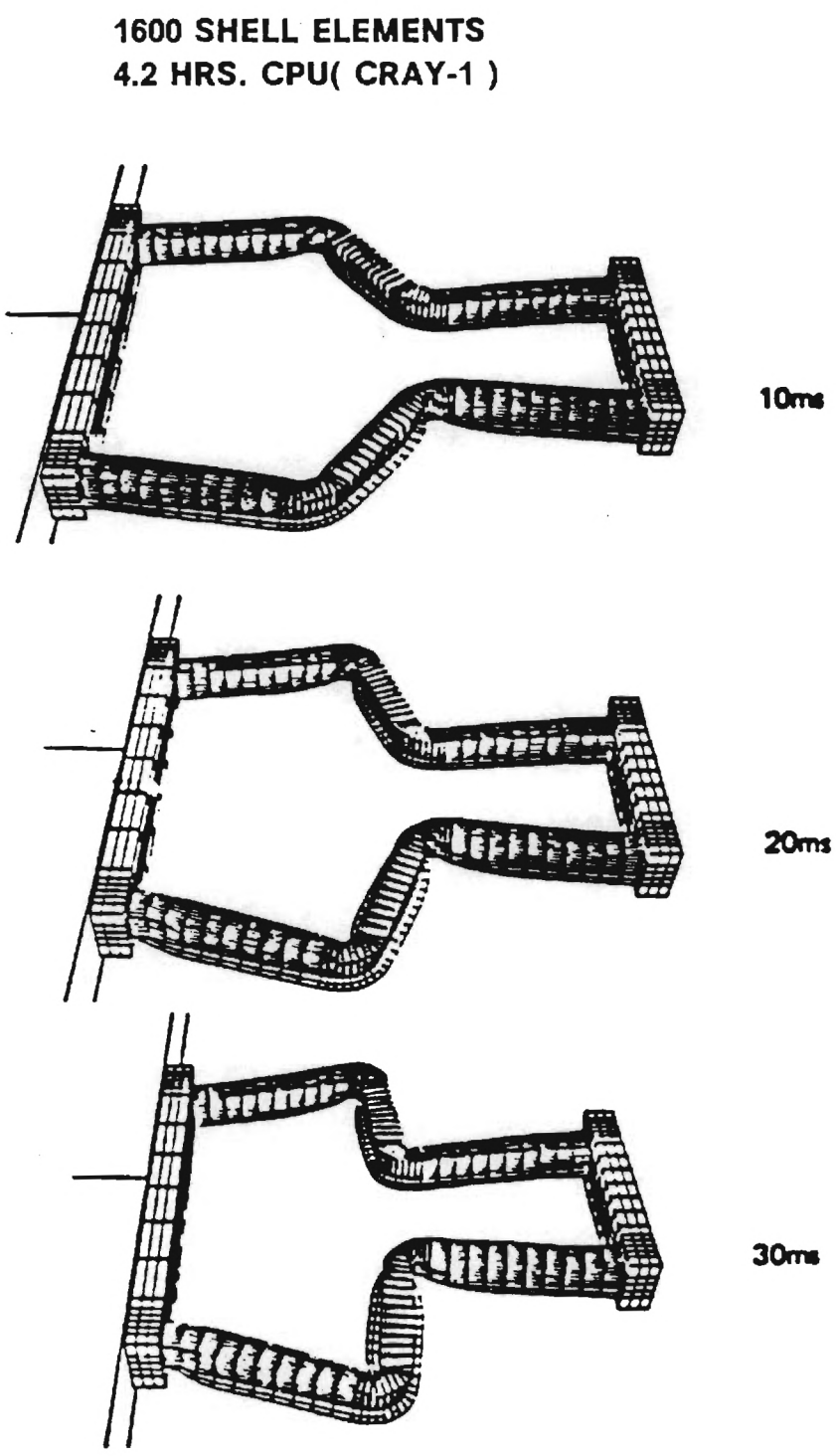
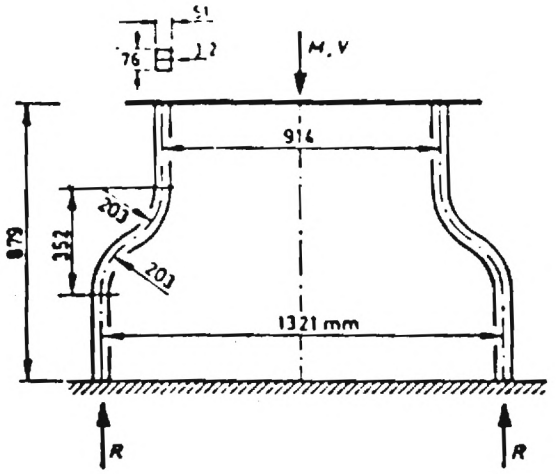


Figure 1. Deformed Shaped at 10ms Output Intervals



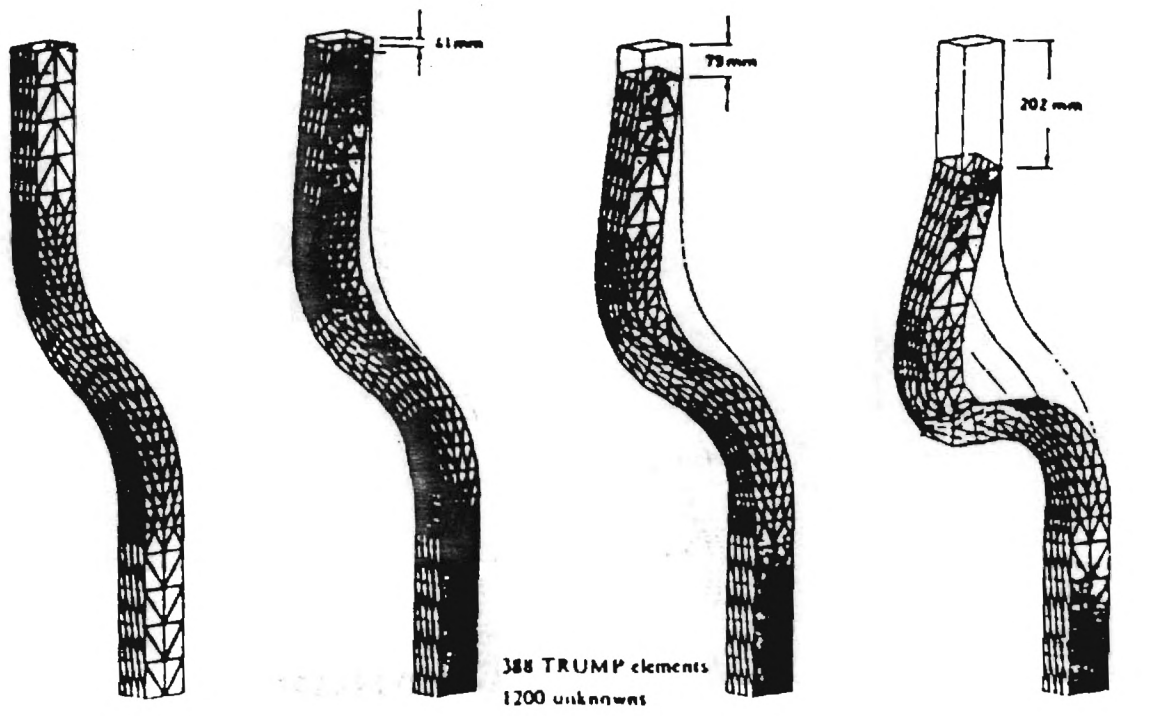
Impact mass $M = 1932$ kg
Impact velocity $V = 4.65$ m/s

Steel material
$E = 2.07 \cdot 10^5$ N/mm²
$\sigma_0 = 250$ N/mm²
$\frac{\sigma}{\sigma_0} = 1 + \left(\frac{\dot{j}}{40}\right)^{0.2}$
$\rho = 7.9$ kg/dm³

Torque box subjected to impact

1.5 HRS. CPU (CRAY-1)

J. ARGYRIS, H. A. BALMER, J. ST. DOLTSINIS AND A. KURZ



t = 0 t = 9 ms t = 17.5 ms t = 45 ms

Figure 2. S-Frame Test Problem

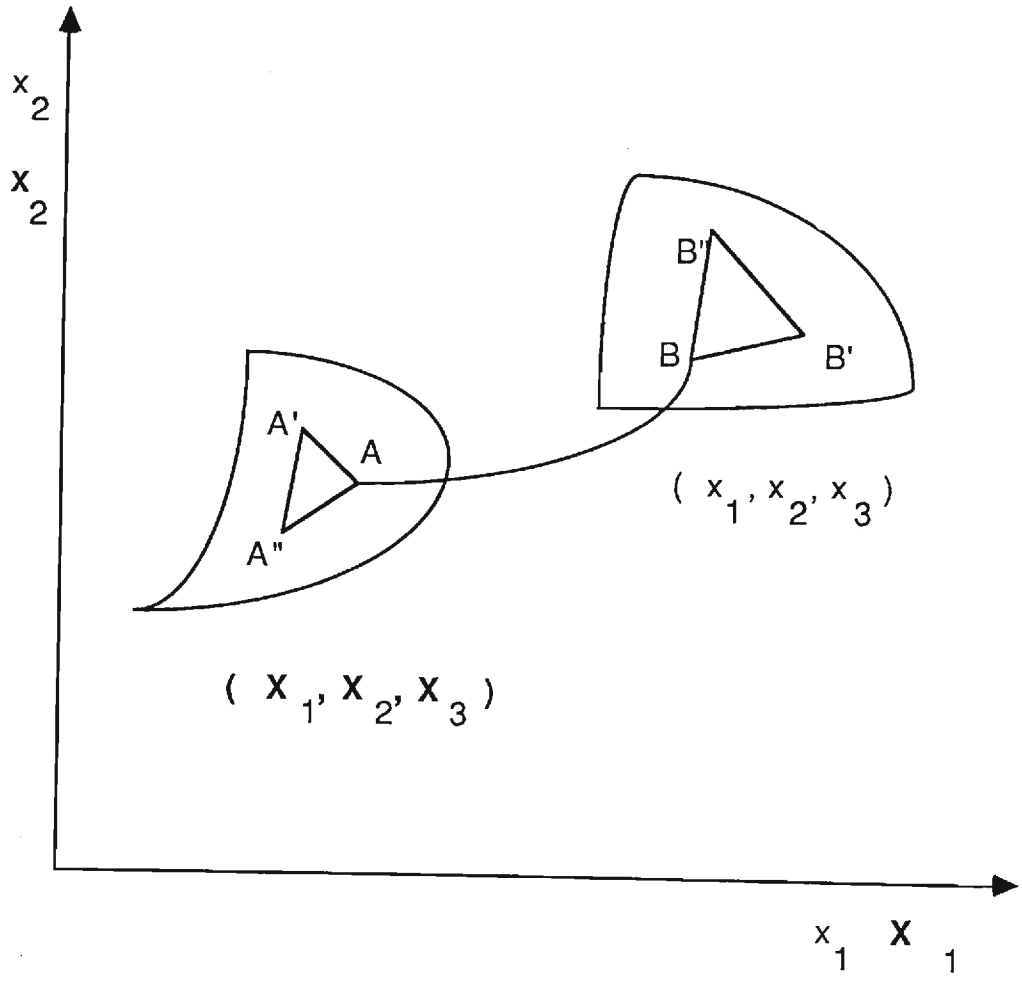


Figure 3. Deformation of a Body

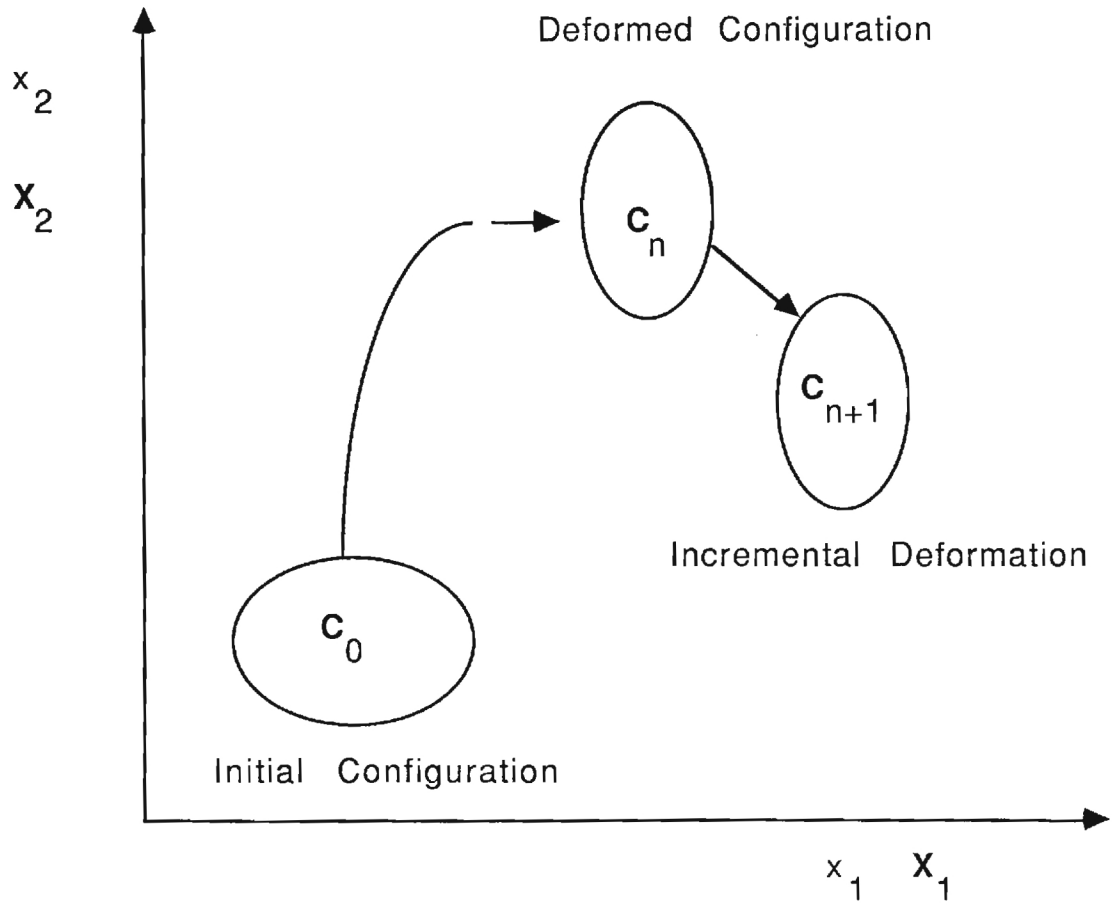


Figure 4. Reference Configurations for Description of Motion

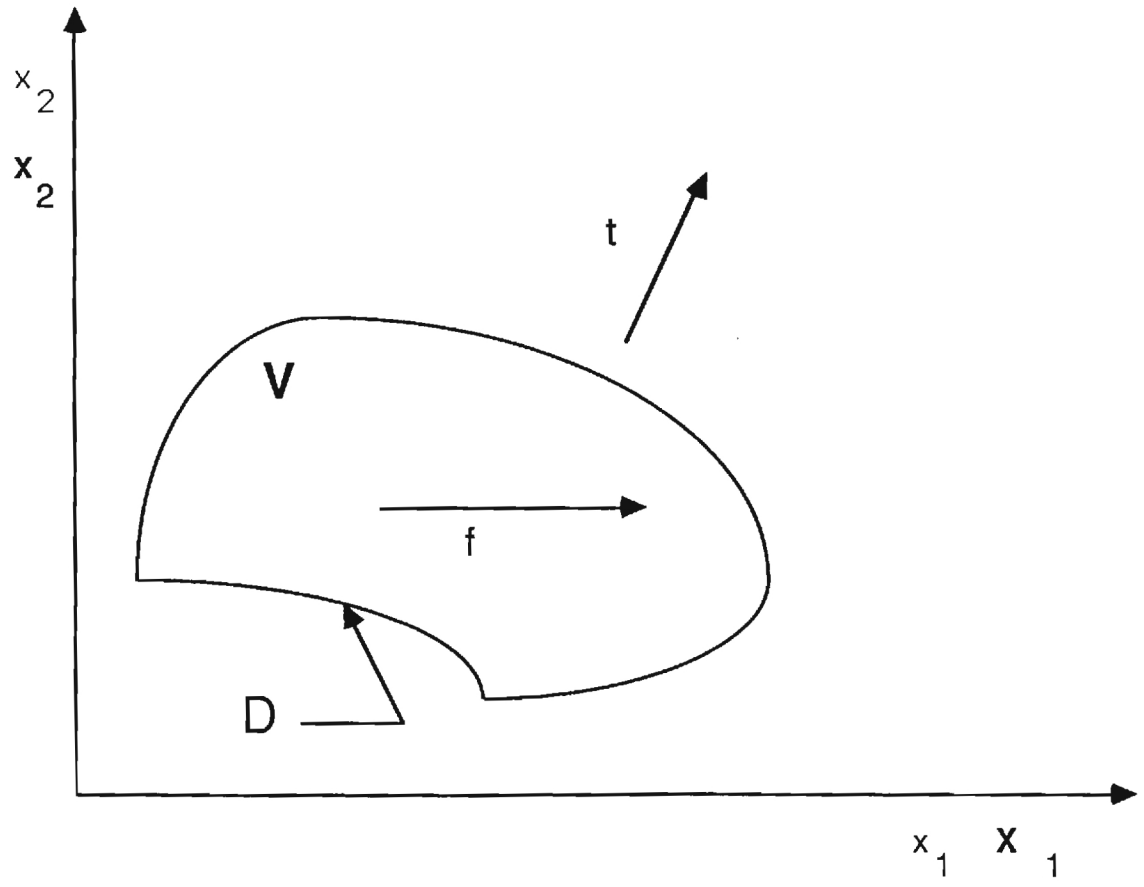


Figure 5. Force Acting on a Body with Volumn V and Surface D

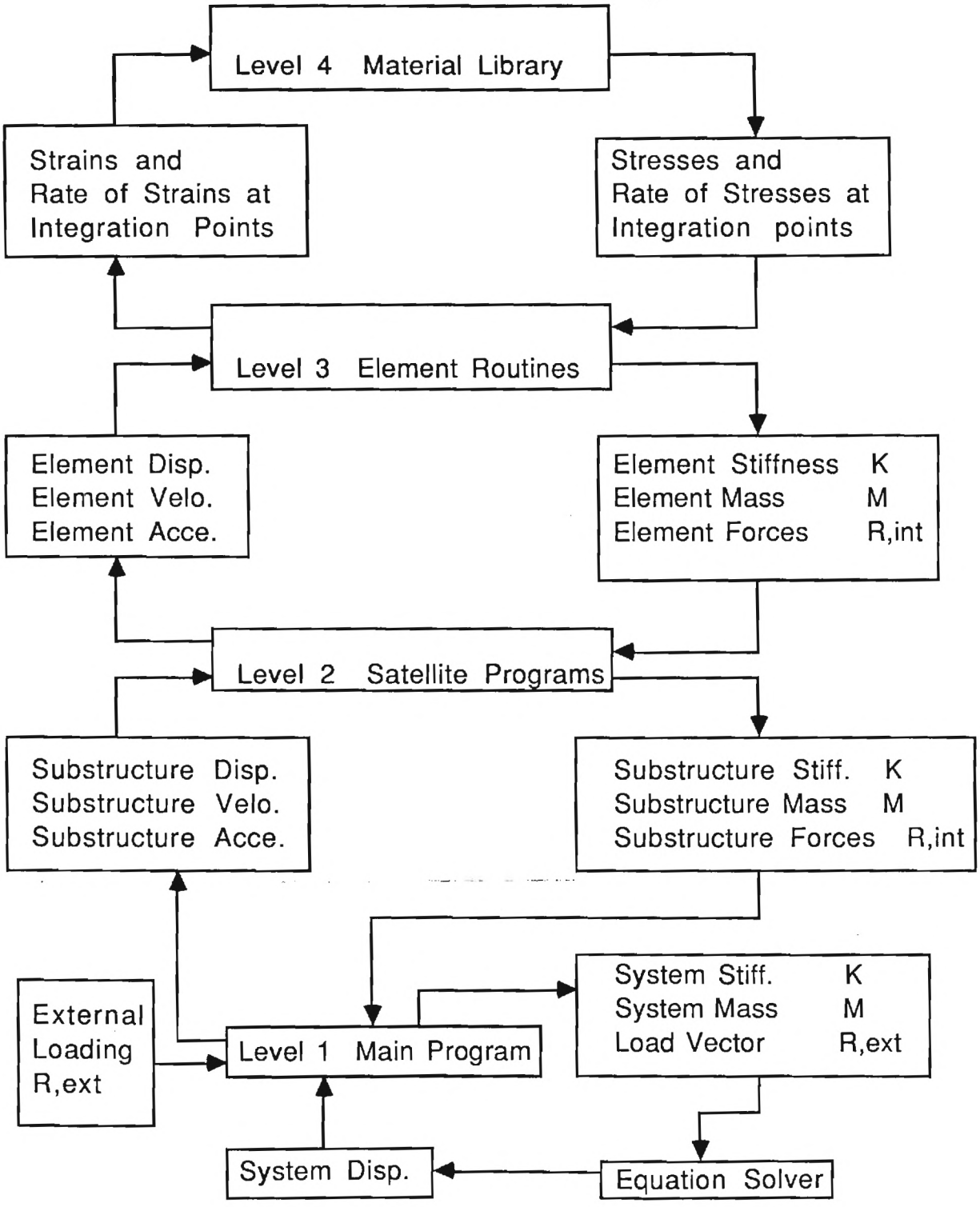


Figure 6. Nonlinear Analysis (FENRIS)

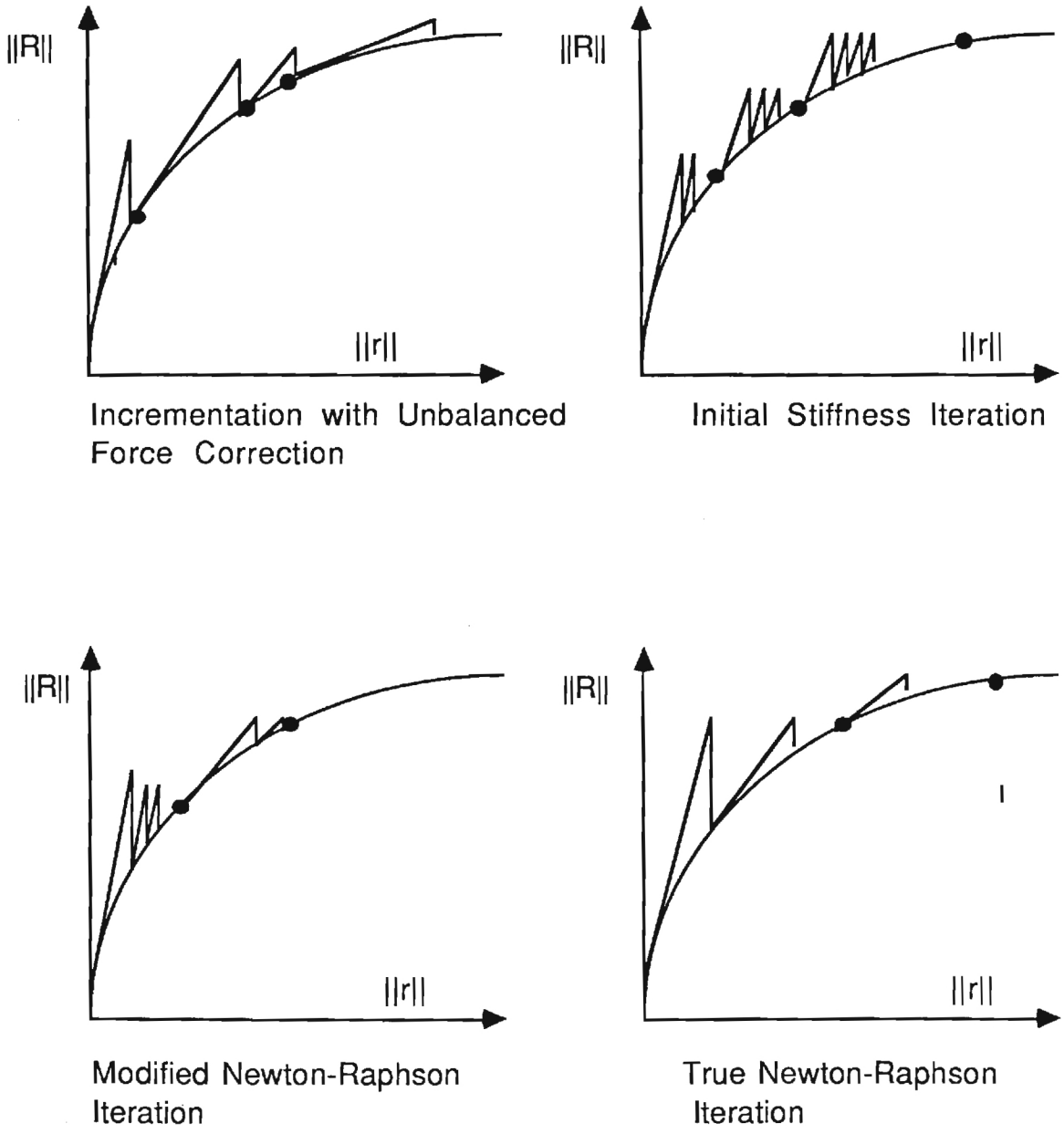


Figure 7. Illustration of the Alternative Solution Methods (FENRIS)

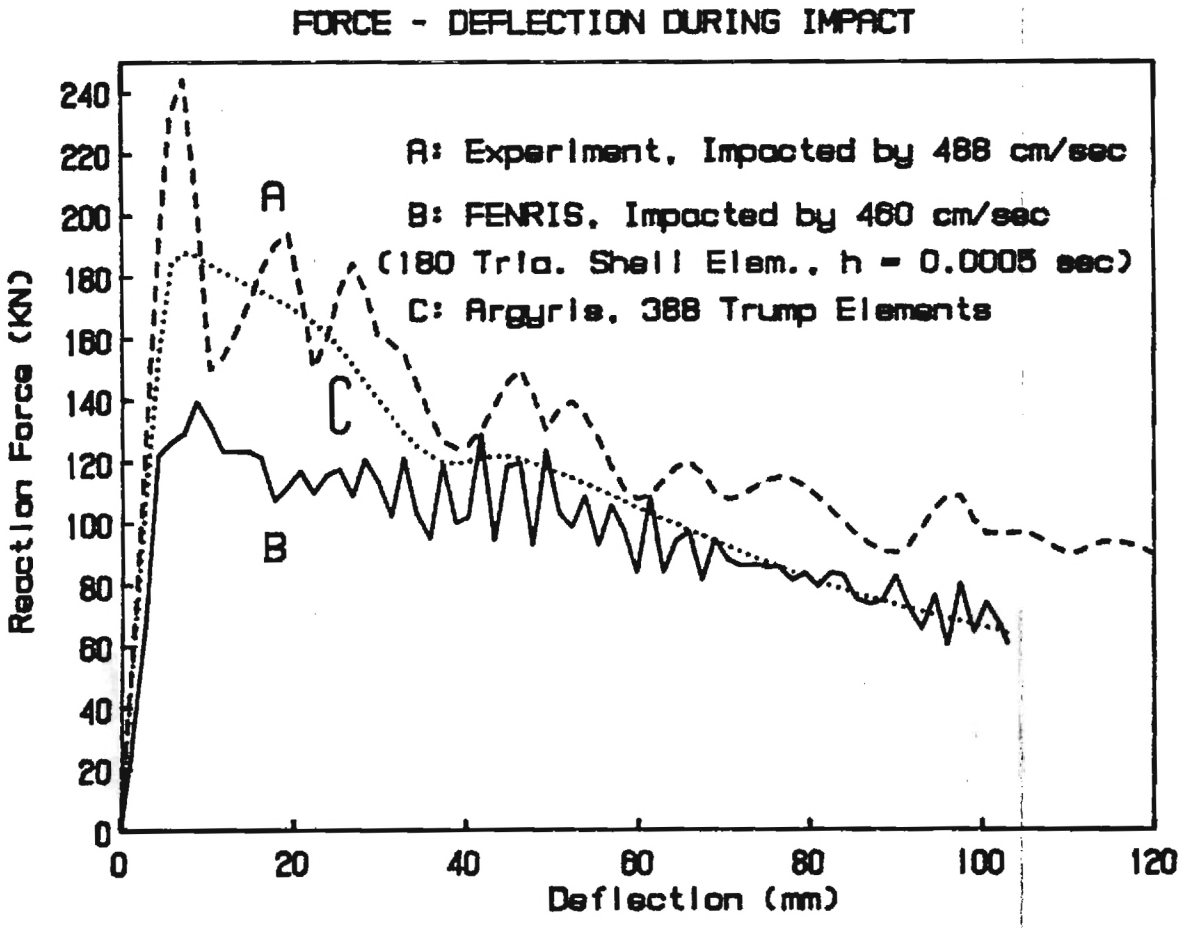


Figure 8. Force - Deflection During Impact

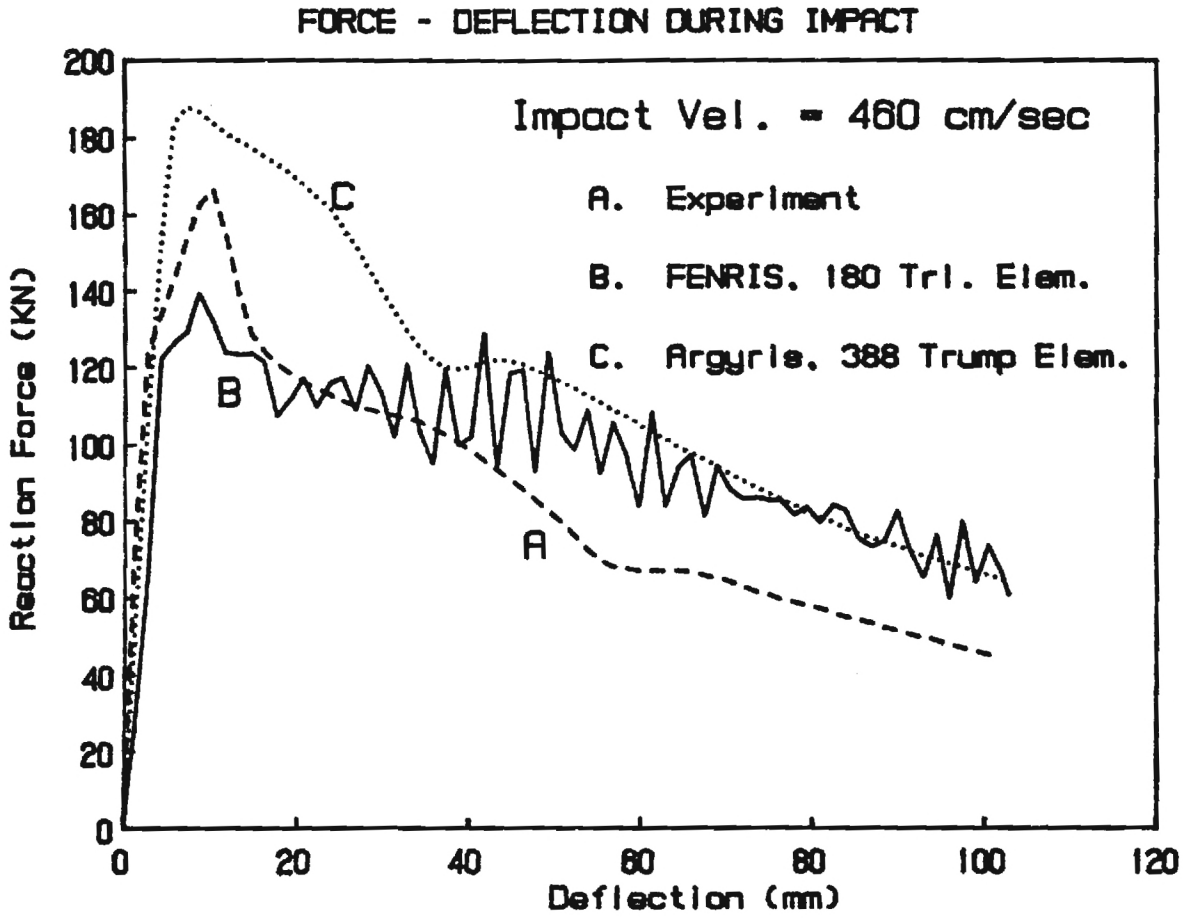


Figure 9. Force - Deflection During Impact

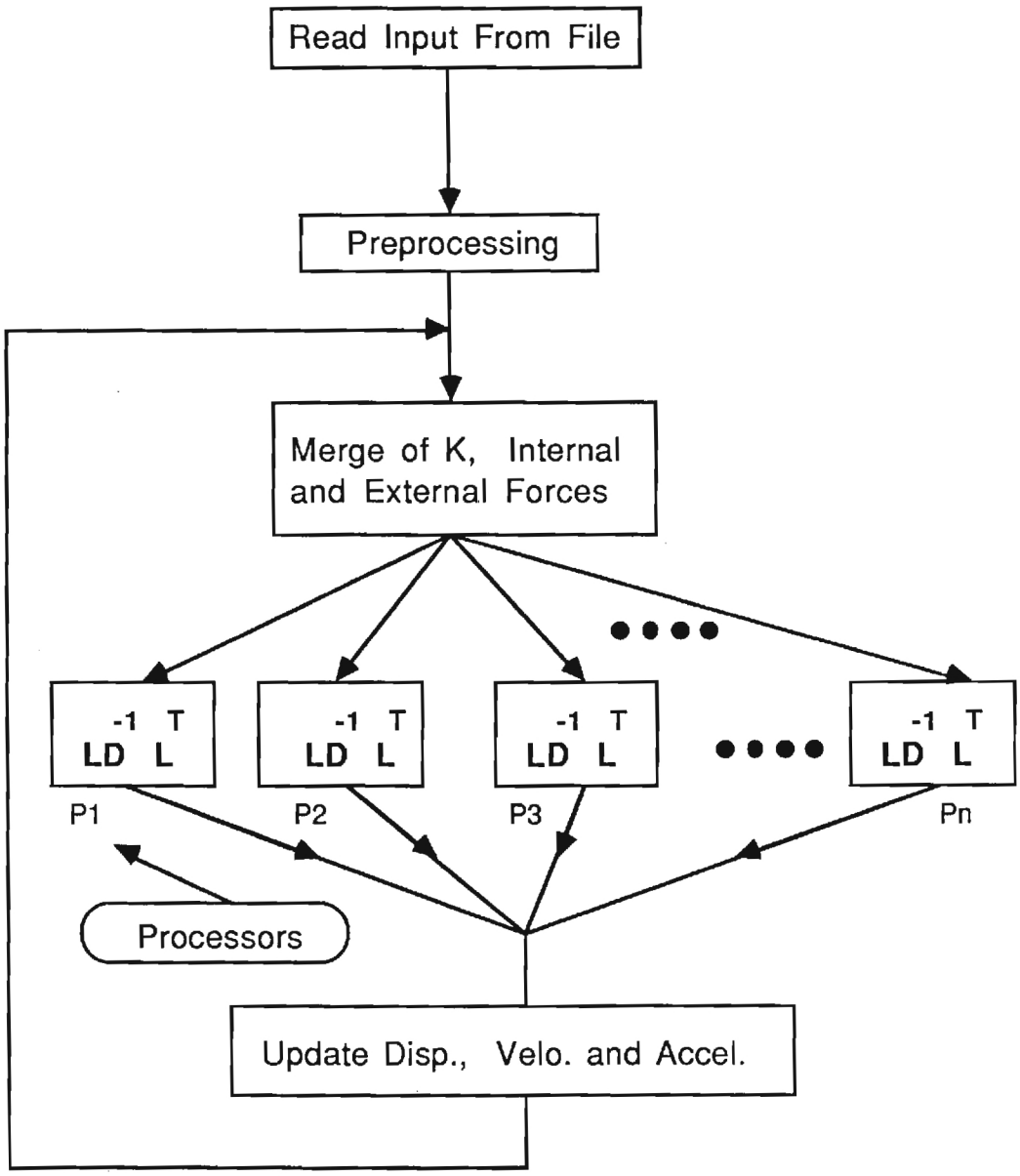


Figure 10. FENRIS/P1

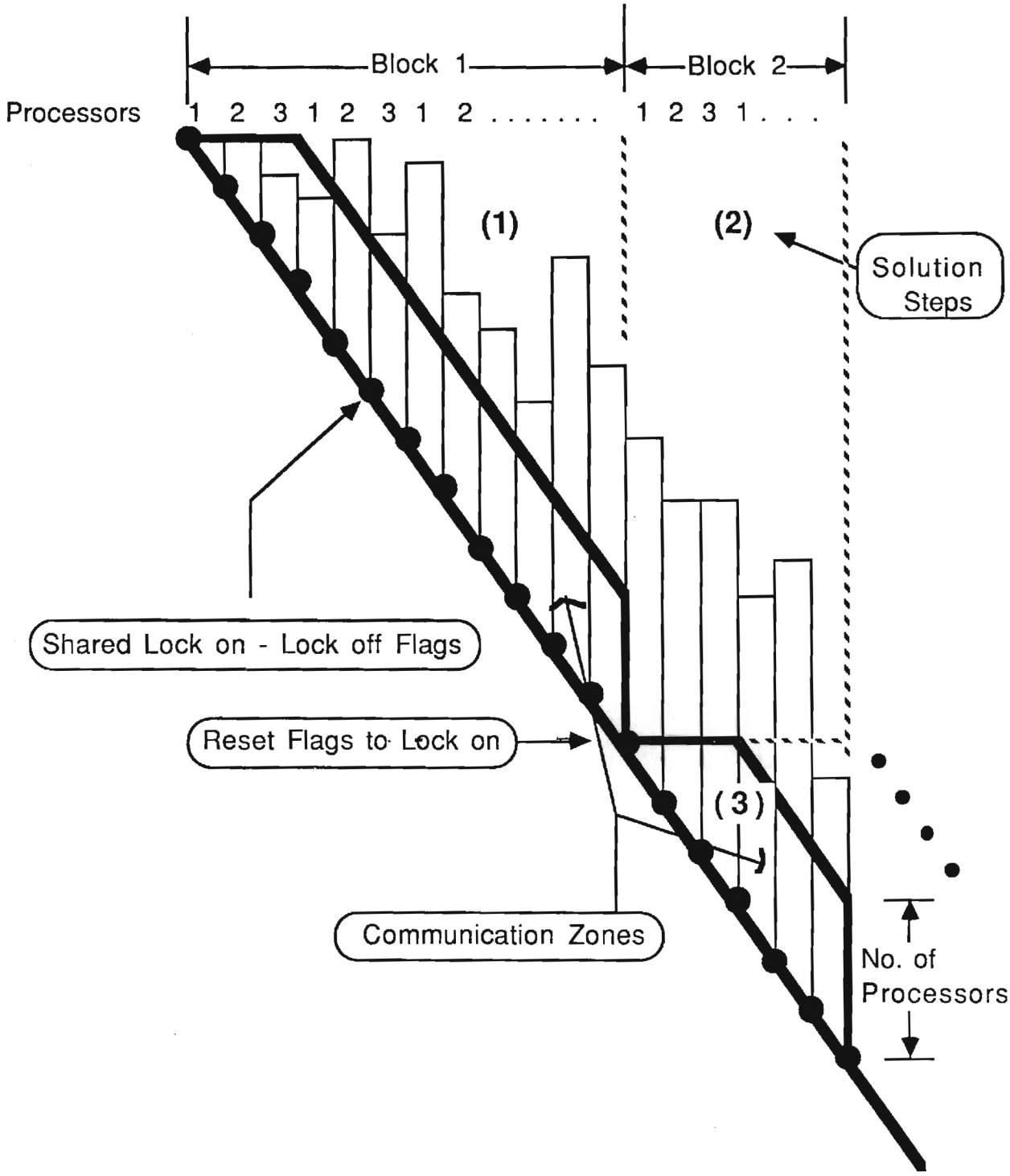


Figure 11. Skyline $LD^{-1}L^{T}$

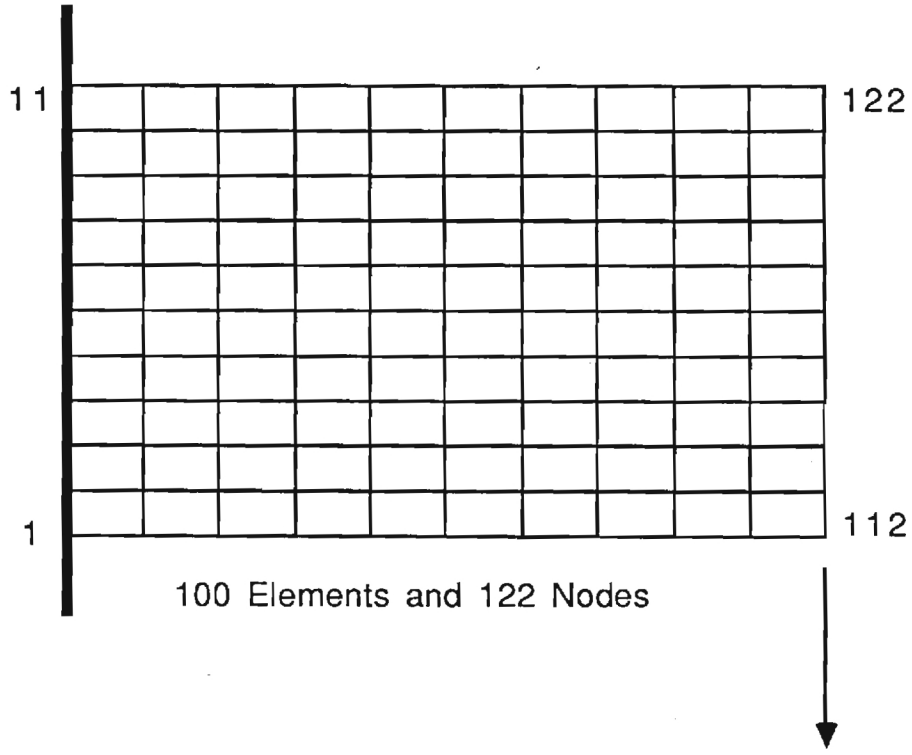


Figure 12. Cantilever Plate (FENRIS/P1)

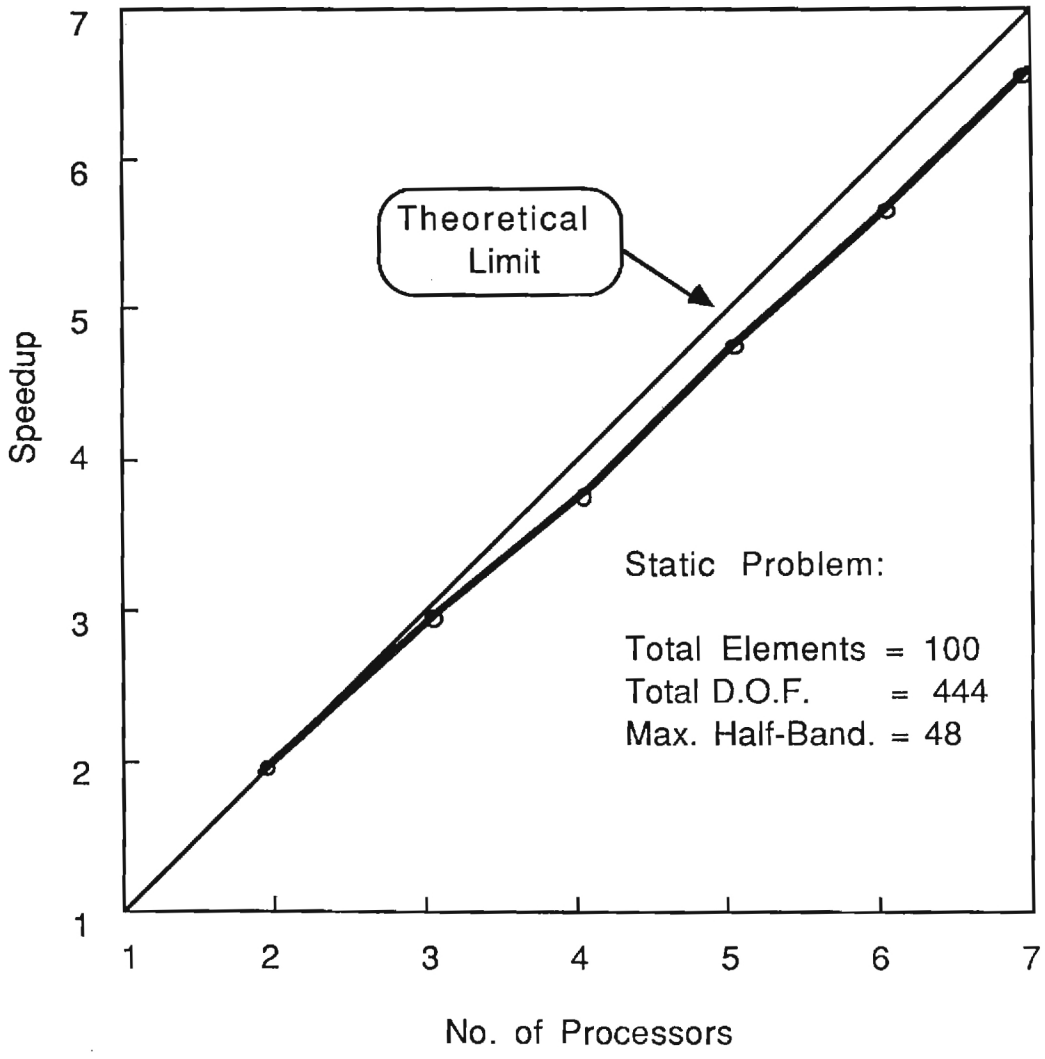


Figure 13. $LD^{-1}L^{T}$ Skyline Decomposition (FLEX/32)

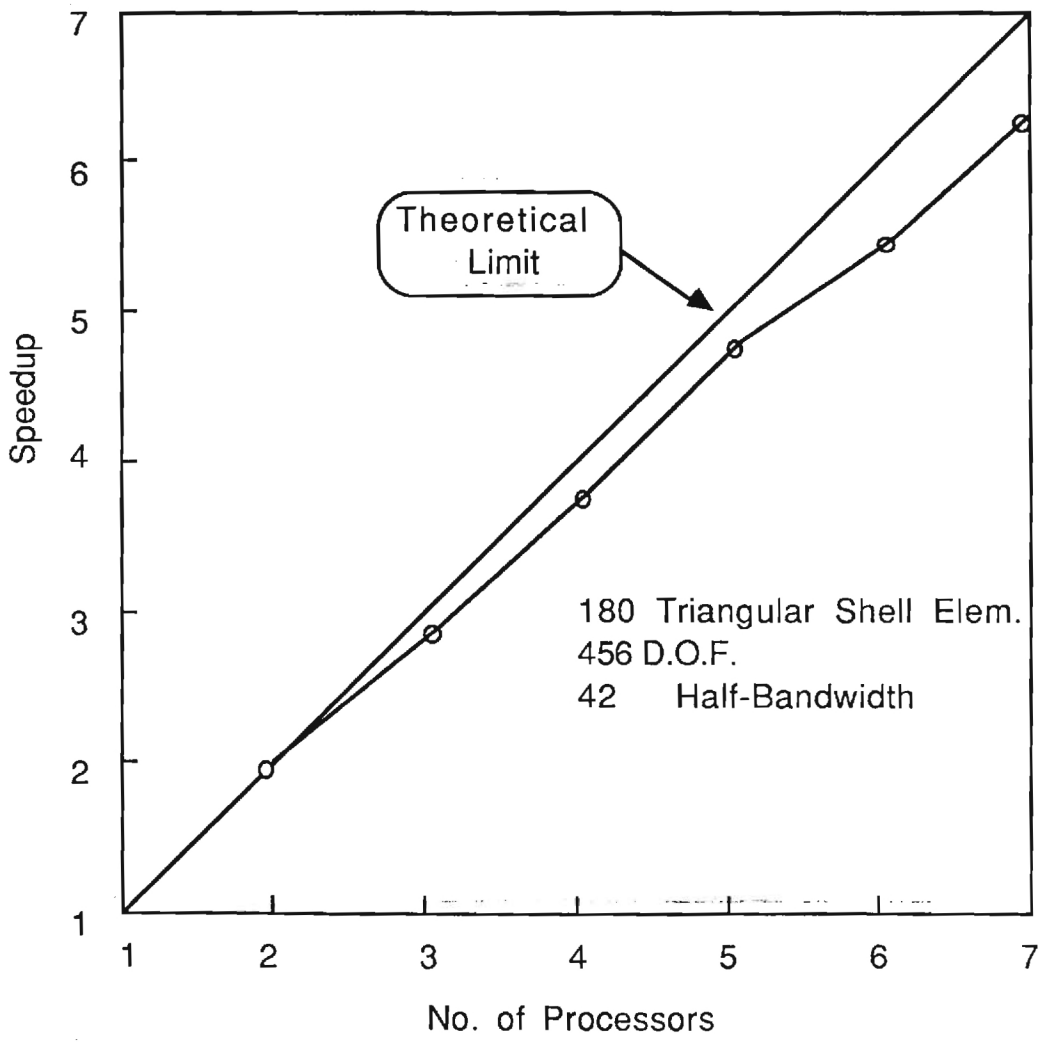


Figure 14. $LD^{-1}L^{T}$ Skyline Decomposition (FLEX/32)

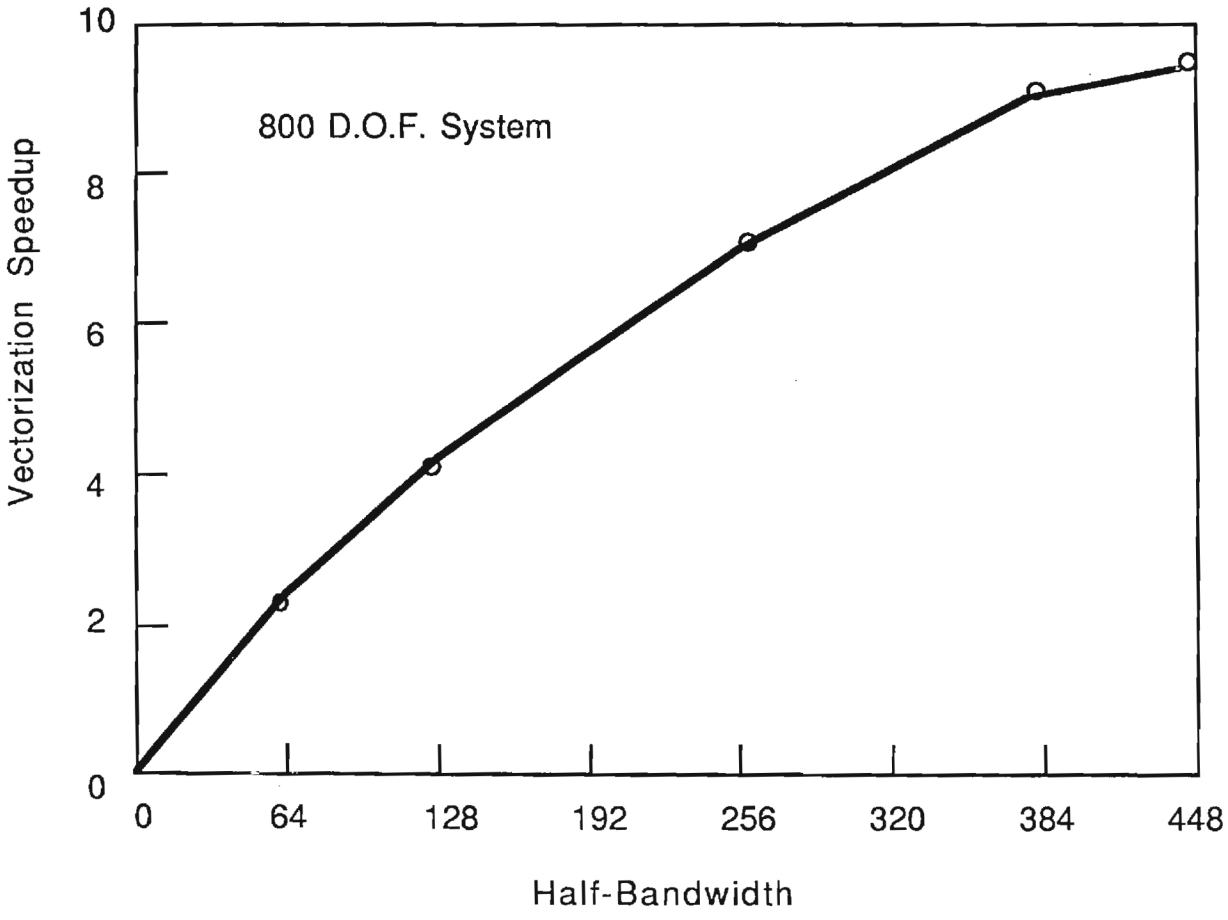


Figure 15. Vector Speedup for $LD^{-1}L^{T}$ Decomposition (CRAY-XMP)

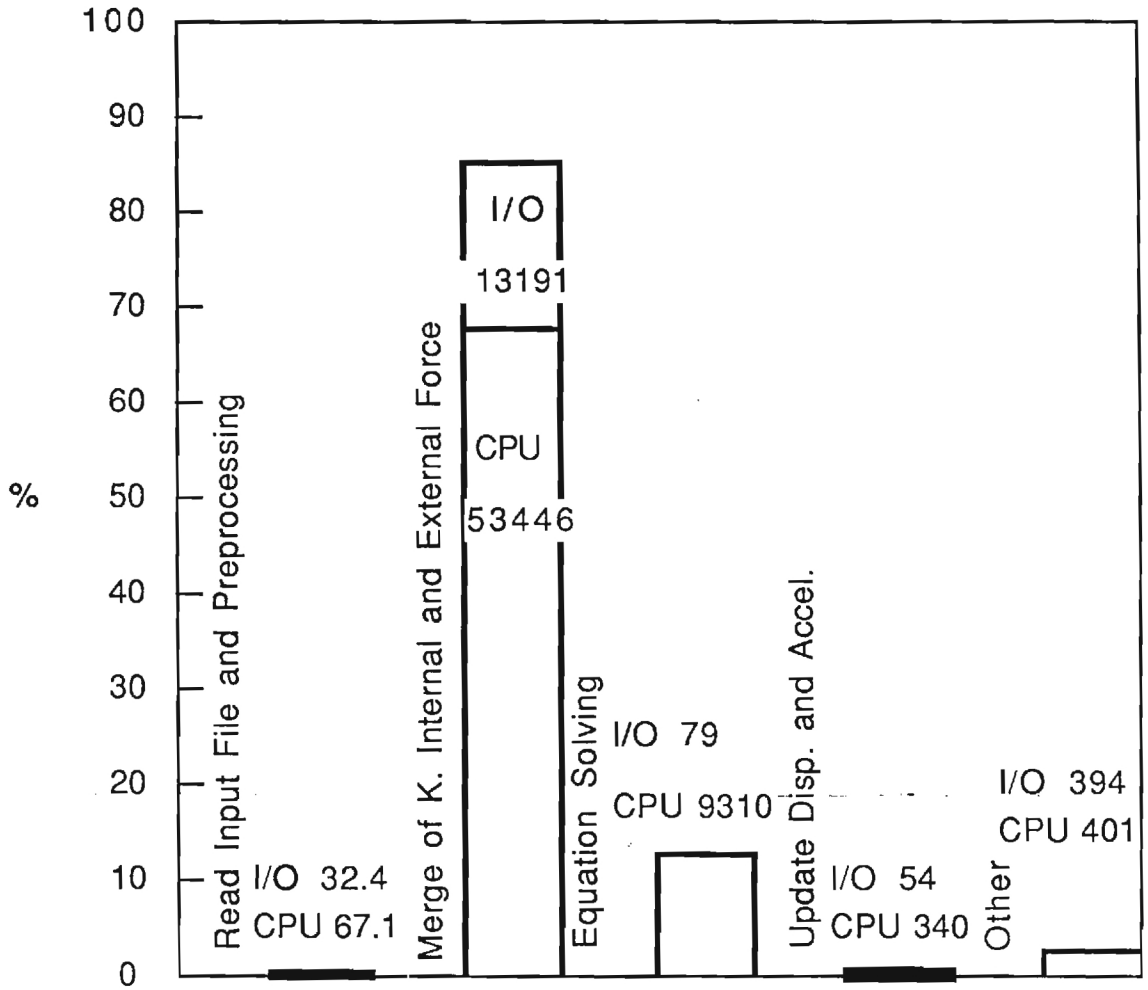


Figure 16. 456 D.O.F S-Frame

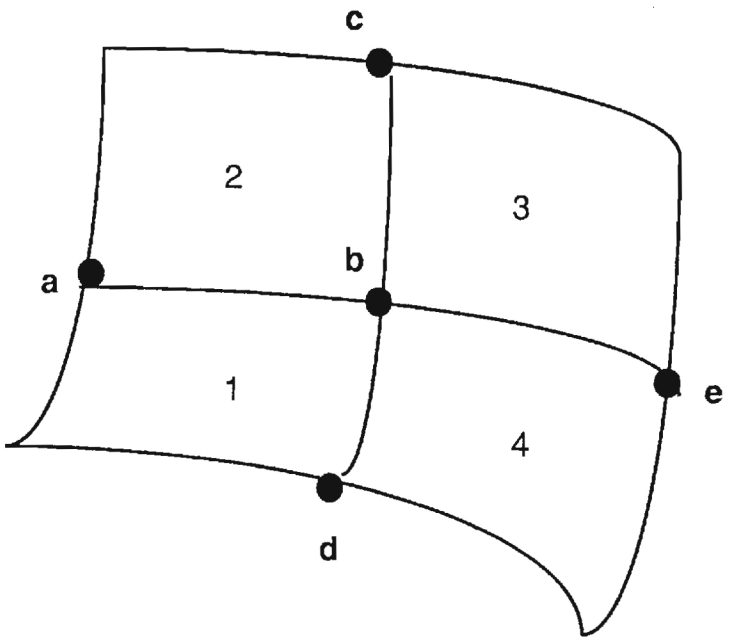


Figure 17. Quadrilateral Element Shared Boundary Nodes

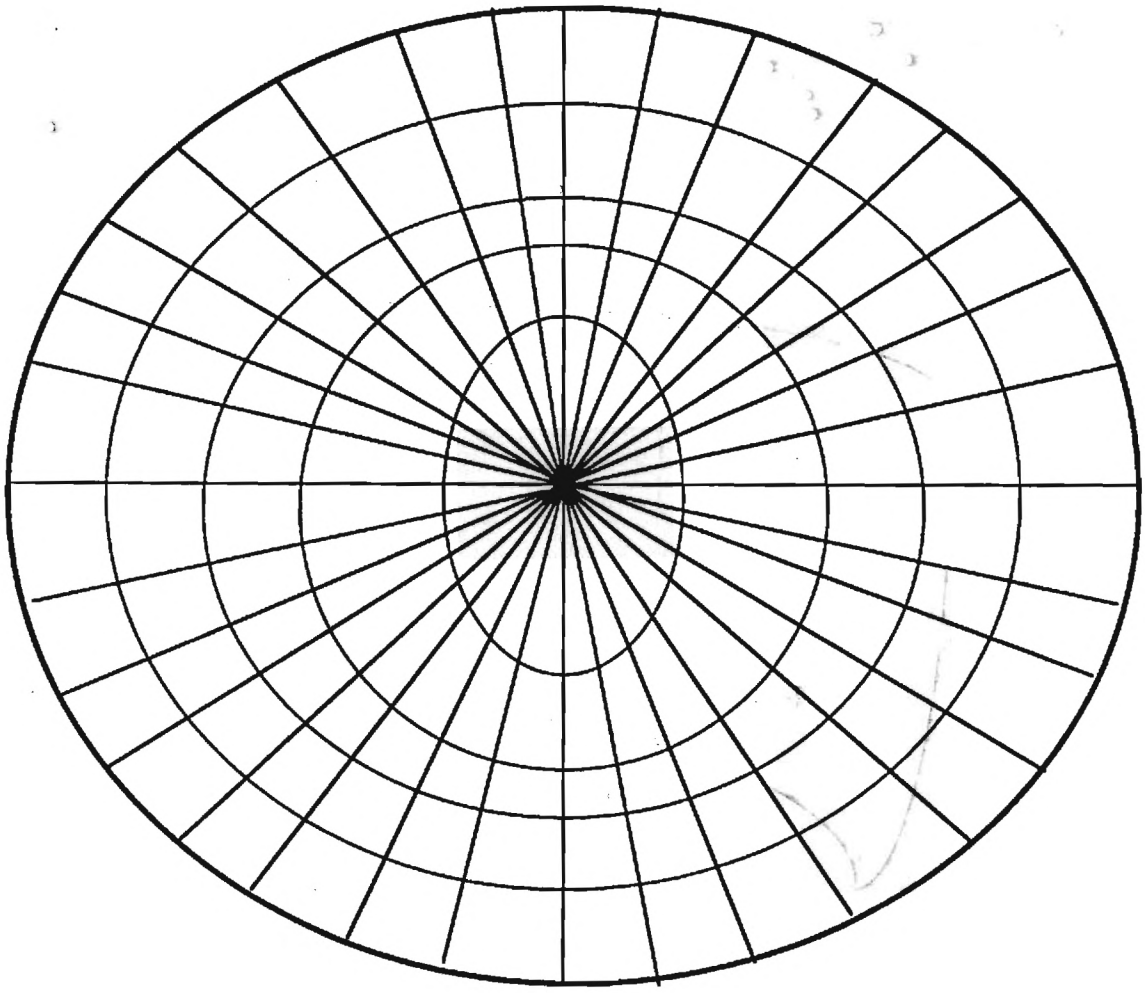


Figure 18. Inefficient Parallel Meshing

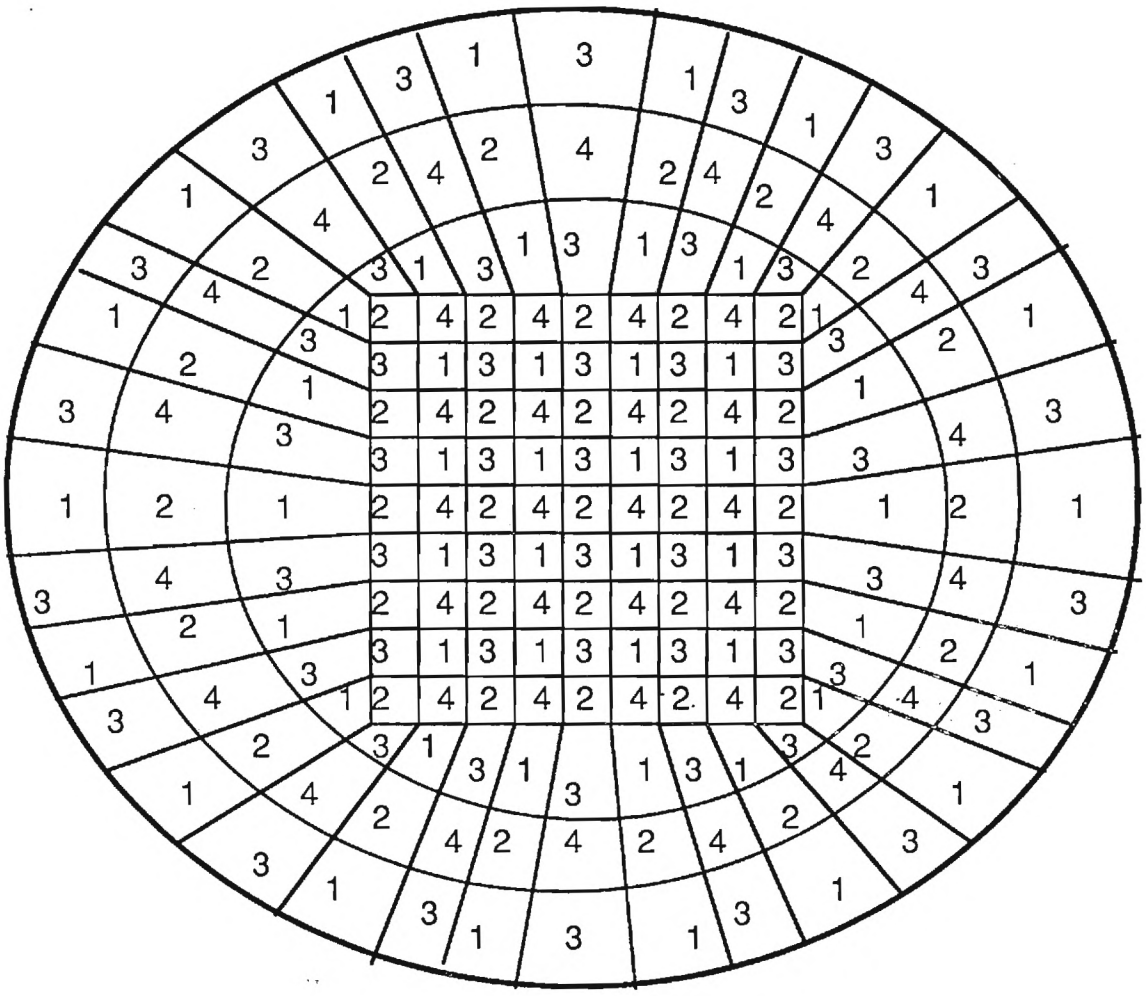
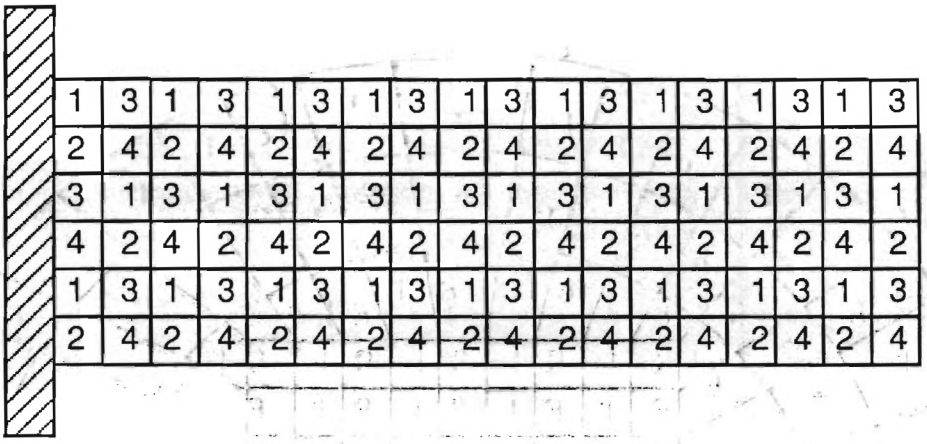


Figure 19. Efficient Parallel Meshing

Option I :



Option II :

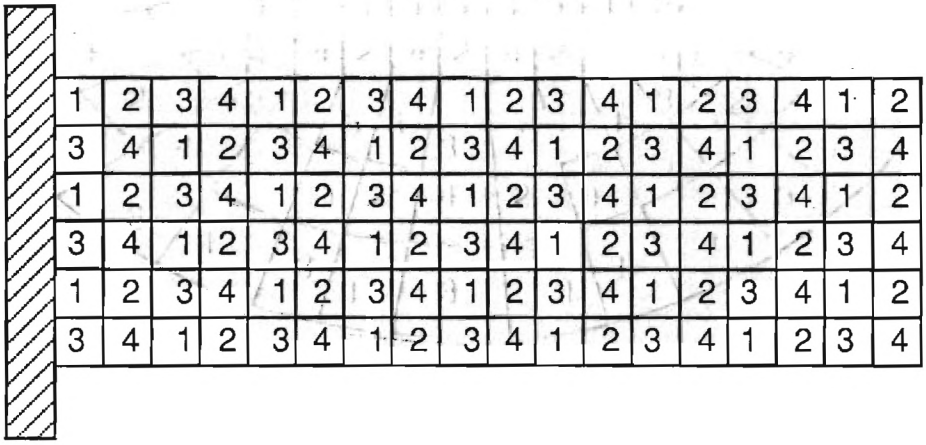


Figure 20. EBE Processors Mapping

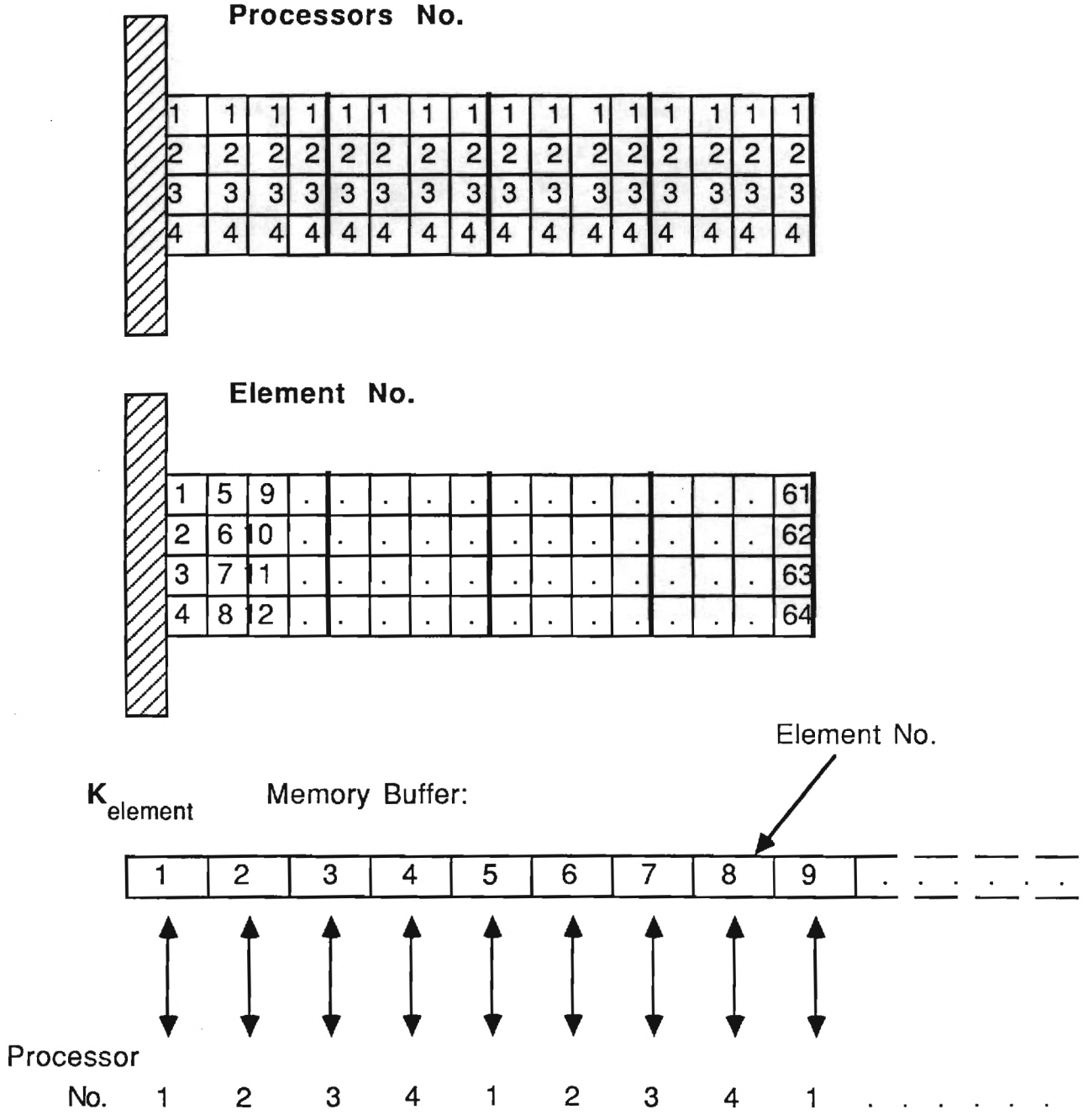


Figure 21. $K_{element}$ Memory Buffer

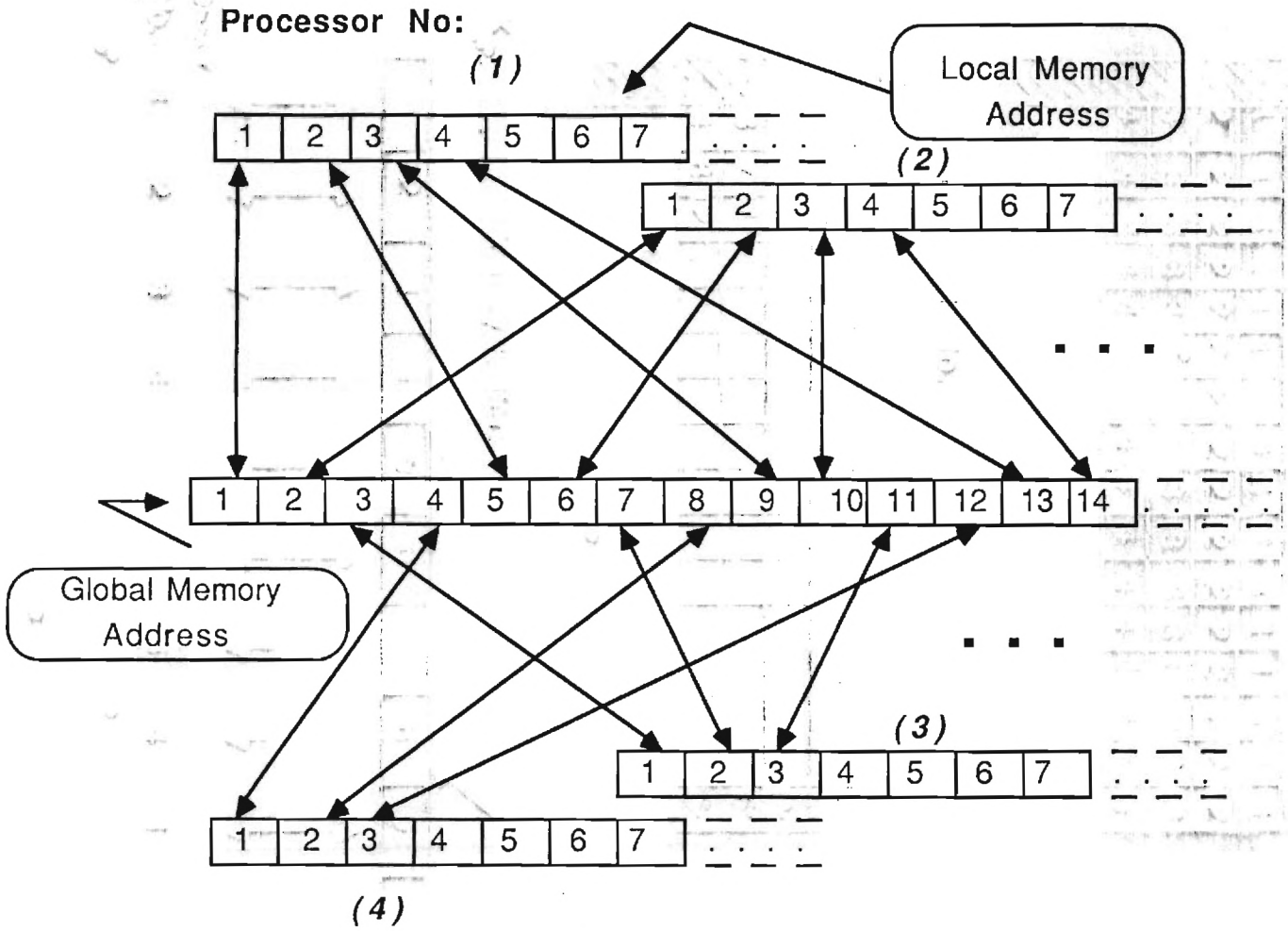


Figure 22. Global-Local Data Mapping

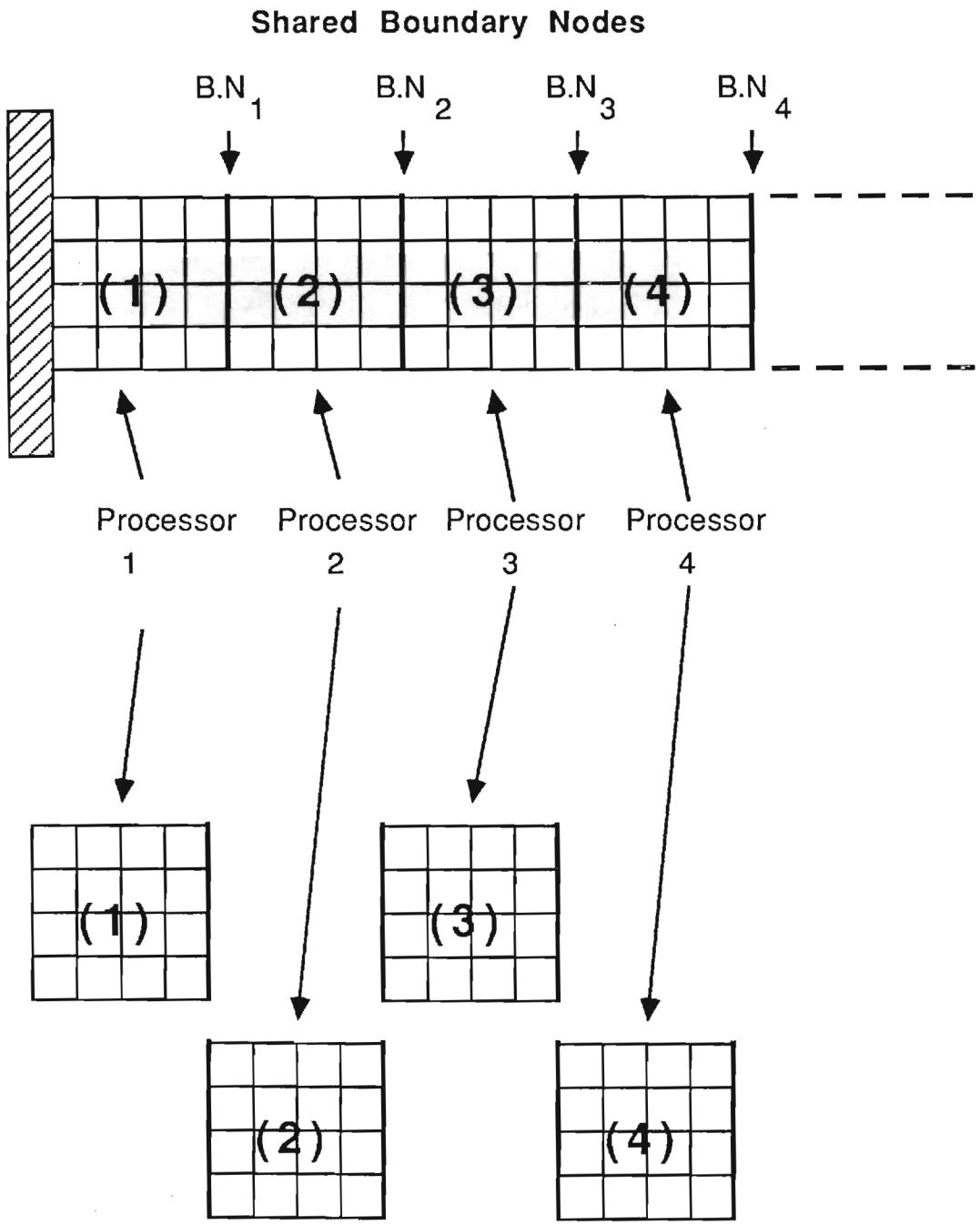


Figure 23. Subdomain Parallelism

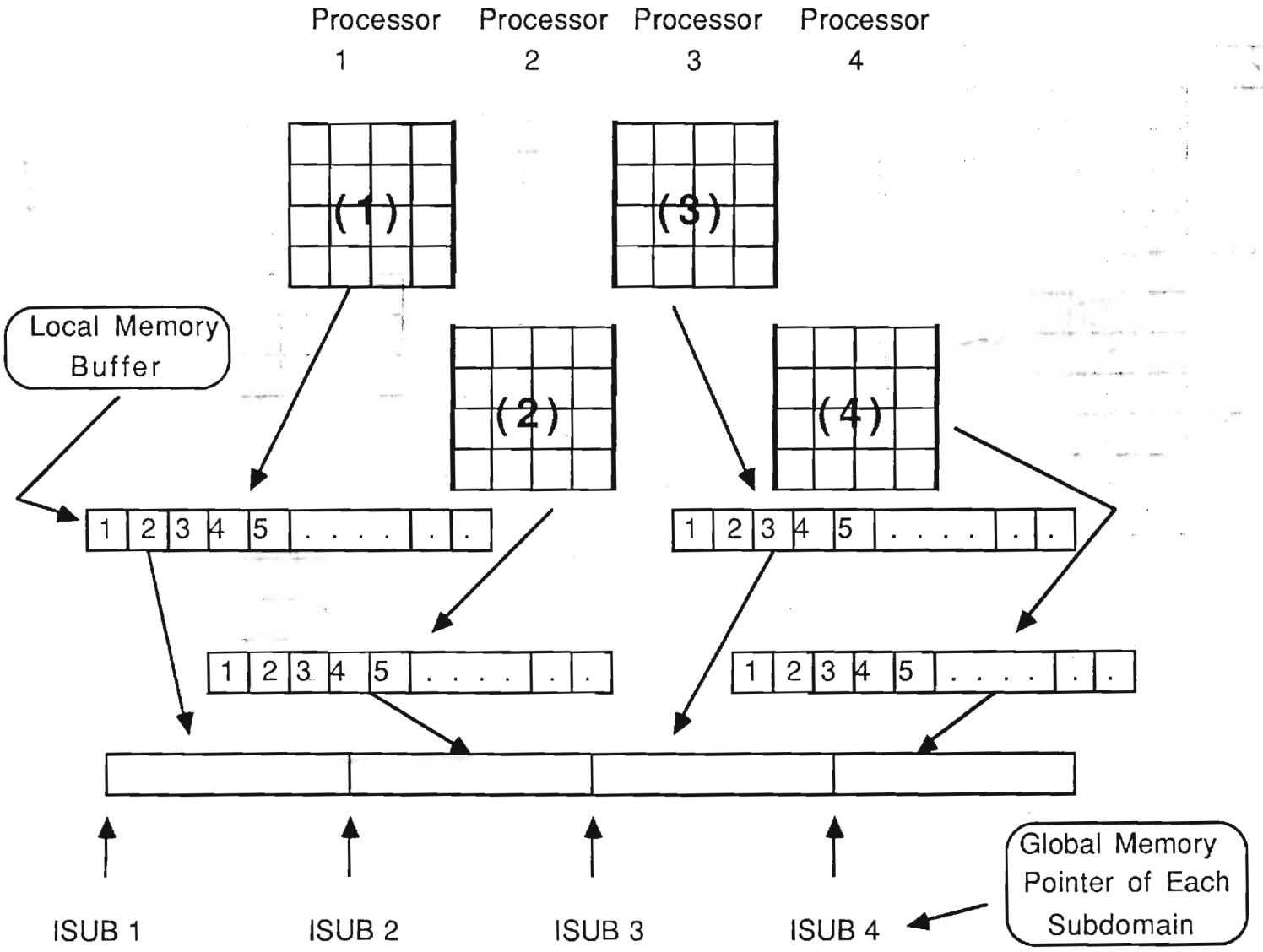
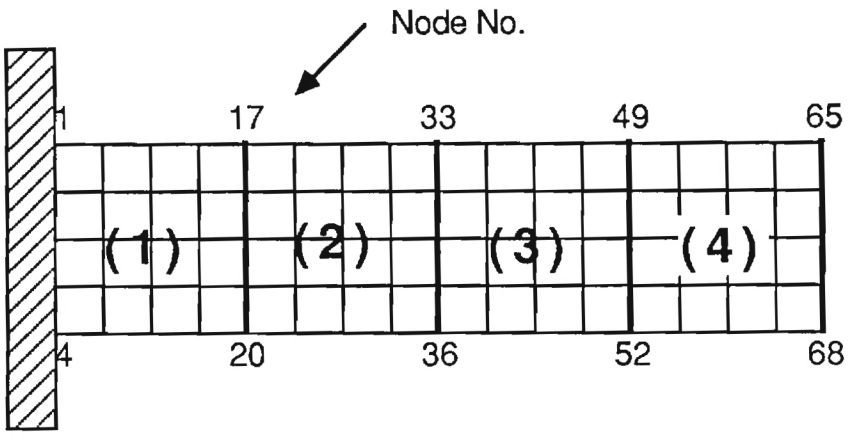


Figure 24. Subdomain Data Mapping



Subdomain	System D.O.F.	Overlap D.O.F
1	1 ~ 96	73 ~ 96
2	78 ~ 192	73 ~ 96 169 ~ 192
3	174 ~ 288	169 ~ 192 265 ~ 288
4	270 ~ 384	265 ~ 288

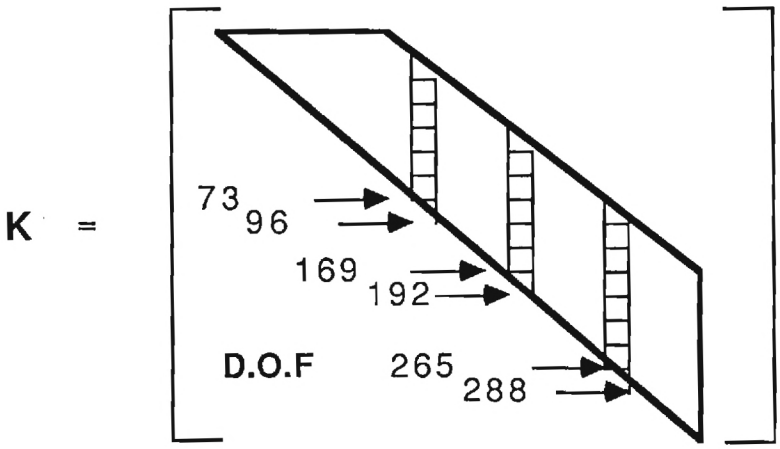


Figure 25. Subdomain Vs. System K Matrix

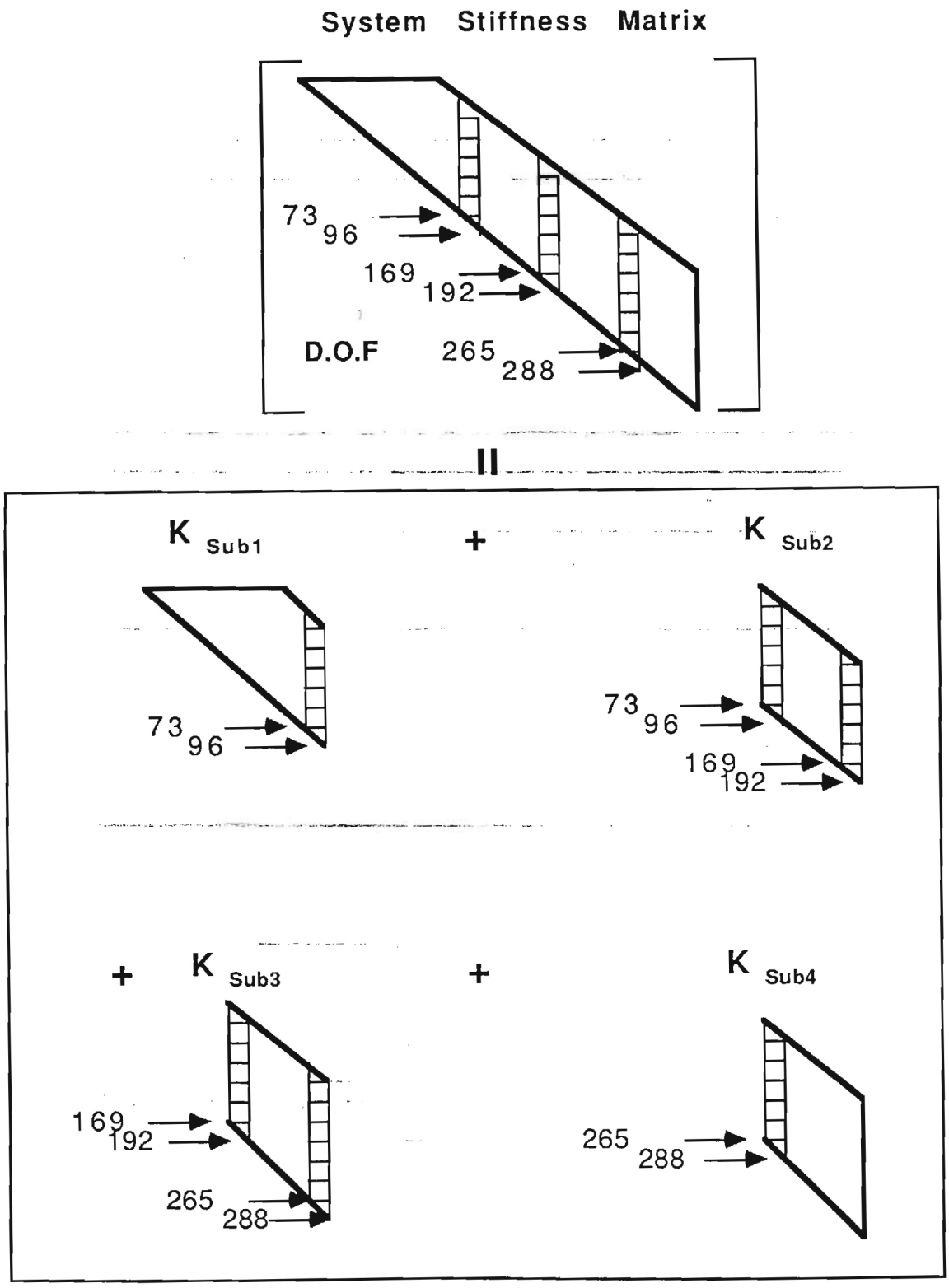
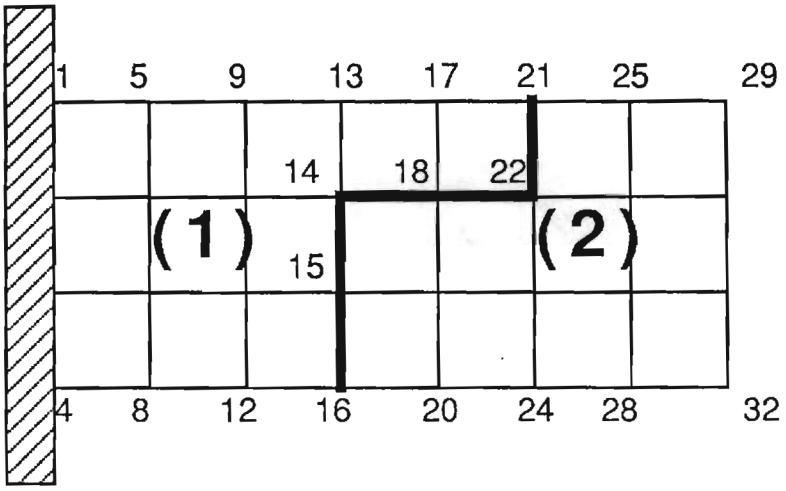
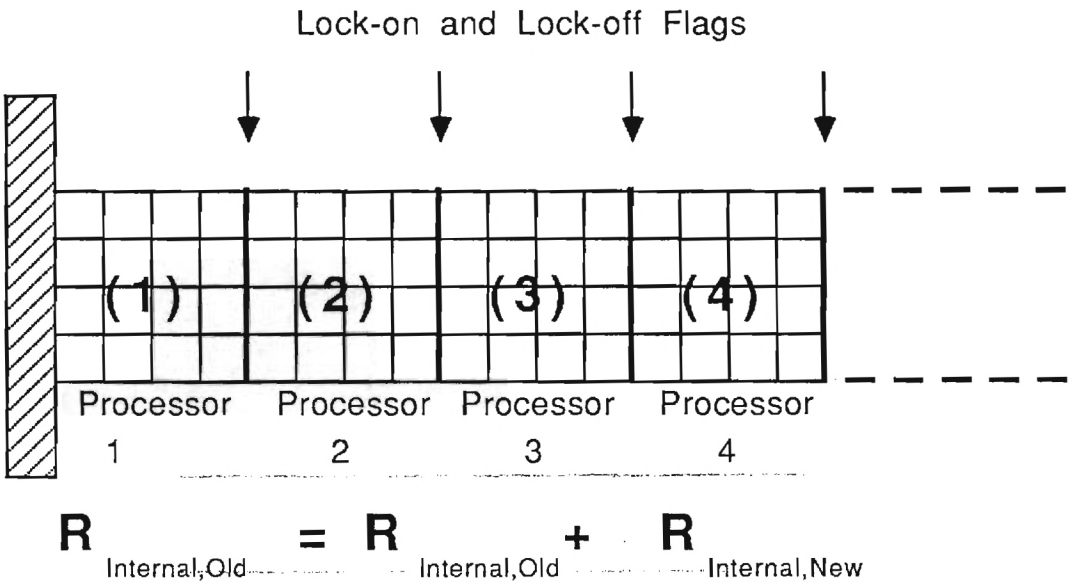


Figure 26. Merge of System Stiffness Matrix



Boundary Nodes	Boundary D.O.F
14 ~ 16	55 ~ 72
18	79 ~ 84
21 ~ 22	97 ~ 108

Figure 27. Irregular Subdomain



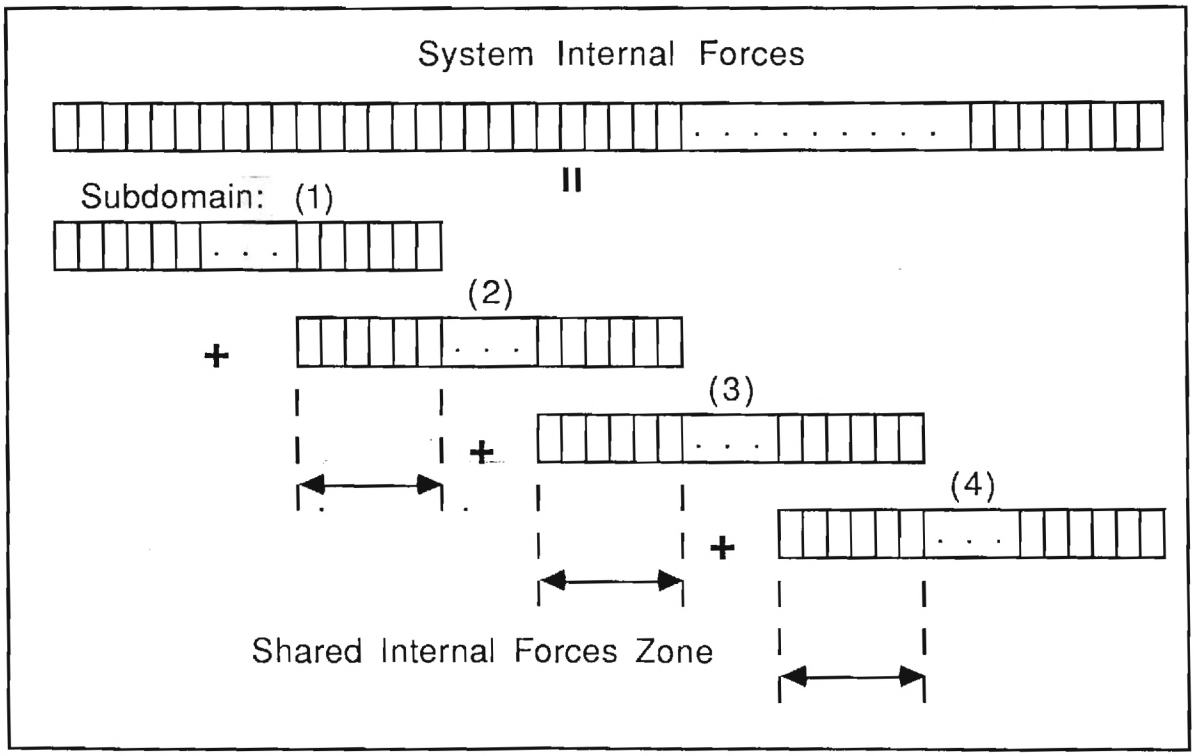
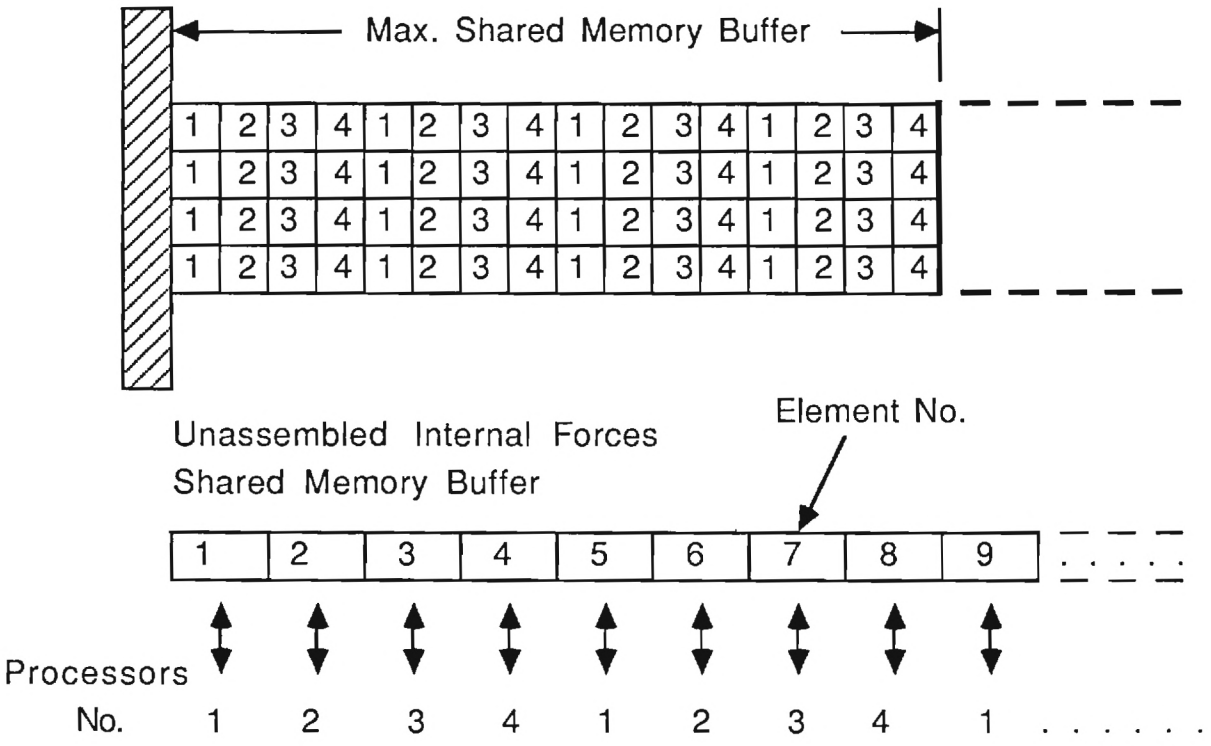


Figure 28. Subdomain Internal Force

EBE Parallelism:



Subdomain Parallelism:

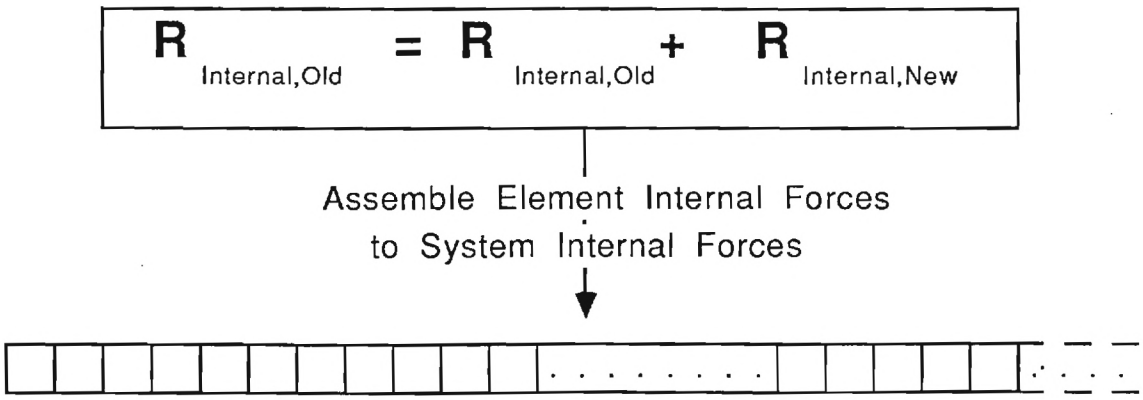


Figure 29. EBE-Subdomain Internal Force

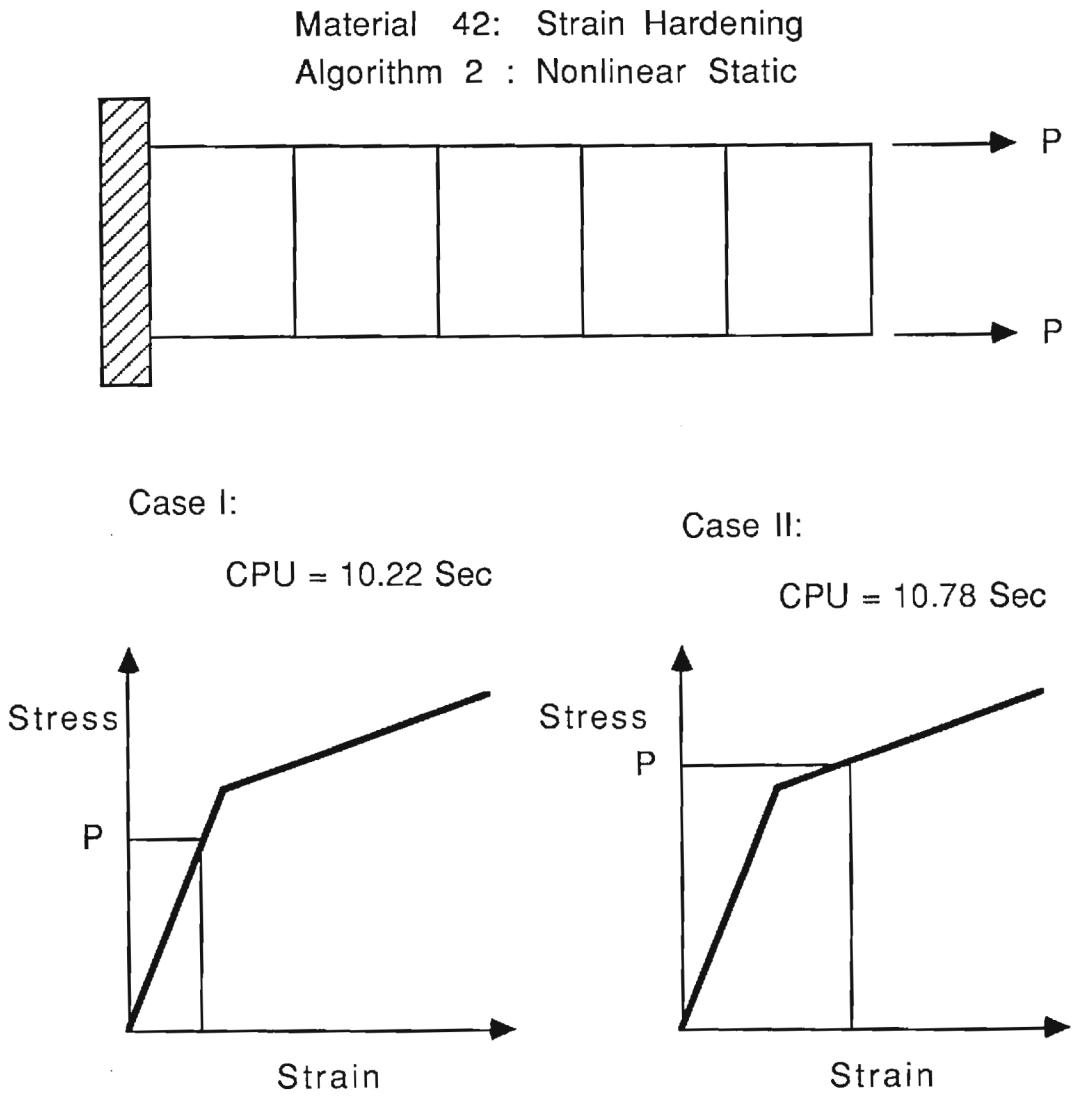


Figure 30. Load Balance Test

FINAL REPORT
PHASE I

COMPUTATIONAL CRASH DYNAMICS METHODS FOR FIFTH GENERATION SUPERCOMPUTERS

By

Robert E. Fulton
Kou-Ning Chiang

Submitted to

General Motors Research Laboratories
Warren, Michigan

December 1986

GEORGIA INSTITUTE OF TECHNOLOGY
A UNIT OF THE UNIVERSITY SYSTEM OF GEORGIA
SCHOOL OF MECHANICAL ENGINEERING
ATLANTA, GEORGIA 30332

1986

COMPUTATIONAL CRASH DYNAMICS METHODS FOR FIFTH GENERATION SUPERCOMPUTERS

by

Robert E. Fulton
Kuo-Ning Chiang

George W. Woodruff
School of Mechanical Engineering
Georgia Institute of Technology
Atlanta, Georgia 30332

Final Report Phase I Submitted to

General Motors Research Laboratories
Warren, Michigan

Georgia Tech Research Corporation E-25-633
to General Motors Research Laboratories

December 1986

COMPUTATIONAL CRASH DYNAMICS METHODS FOR FIFTH GENERATION SUPERCOMPUTERS

INTRODUCTION

The structural design/analysis of future complex engineering products such as automobiles, trucks and aerospace vehicles subjected to crash impact conditions requires significantly greater analytical complexities than many other design conditions. For example, crash design can include the combined effects of dynamic loads, composite materials, multidisciplinary interactions, three-dimensional geometry and nonlinear behavior. Realistic structural models for such designs typically imply large-order finite element or finite difference models and excessively large computational requirements. Since the advent of NASTRAN (ref. 1), general-purpose finite element structural analysis computer programs (e.g., ref. 2) have provided the capability to address a wide range of structures problems including design, optimization, nonlinear and dynamic capabilities. However, when these programs are applied to crash dynamics problems, more computing effort is required than is available with current sequential computers. Moreover, sequential computers will be inadequate to support analysis requirements for many future engineering designs where effective speeds greater than 10^3 MFLOPS (million floating point operations per second) will be required. For example, calculations for such problems as nonlinear dynamic response, multidisciplinary interactions and optimum design are all severely limited by computation speed, and models are usually restricted to small-order problems or highly simplifying approximations.

Projected advances in computer technology indicate that significant increases in effective calculation speed will be available in the 1990's, through fifth generation supercomputer architectures consisting of arrays of processors operating in parallel on different tasks (see e.g., ref. 3 for a survey). Such advanced supercomputers, denoted at MIMD (multiple instruction, multiple data) computers, have the potential for increasing effective calculation speeds by several orders of magnitude. But this potential increase in speed cannot be effectively utilized without the development and implementation of appropriate numerical algorithms for structures which take advantage of the parallel computation features of this new generation of computers. Use of existing conventional algorithms and software will not realize the full potential of these new MIMD computers, and research is needed in the development of parallel structural analysis/design algorithms for these computers (ref. 4-19). This research has been to develop and evaluate parallel methods for crash dynamic analyses of complex nonlinear finite element and/or finite difference structural problems.

Issues involved in implementation of parallel structural algorithms on MIMD supercomputers are much more complex than for current sequential computers, and the total hardware/software system must be taken into consideration. The implementation criteria that influence the efficiency of an algorithm include the amount of computation versus the number of processors, the communication paths and synchronization delays, and the size of a problem in relation to the number of processors used. The flow of the algorithm must be analyzed to identify those calculations which must be sequential and those which can be done in parallel. Tasks are partitioned onto a processor array so that the communication paths are most effective. Efficient algorithms typically maximize parallel and minimize sequential calculations.

PARALLEL NONLINEAR DYNAMIC RESPONSE

The finite element equations governing the nonlinear crash dynamic response of a structure take the form

$$\ddot{MU} + \dot{CU} + KU + G(U) = F(t) \tag{1}$$

where M, C, and K are the mass, damping and stiffness matrices, U the displacement vector, F the forcing function and G(U) represents the nonlinear effect. Dots denote differentiation with respect to time t. Consider the integration of these equations from time step j to j+1 over the time interval k using, for example, the Newmark trapezoidal integration formula (see ref. 20 with $\alpha = 1/2$, $\beta = 1/4$)

$$\dot{U}_{j+1} = \dot{U}_j + \frac{k}{2} [\ddot{U}_j + \ddot{U}_{j+1}] \tag{2}$$

$$U_{j+1} = U_j + \frac{k}{2} [\dot{U}_j + \dot{U}_{j+1}]$$

$$= U_j + k \dot{U}_j + \frac{k^2}{4} [\ddot{U}_j + \ddot{U}_{j+1}] \tag{3}$$

Writing equation (1) at station j+1 gives

$$\ddot{MU}_{j+1} + \dot{CU}_{j+1} + KU_{j+1} + G(U_{j+1}) = F(t_{j+1}) \tag{4}$$

When solving equations (2), (3), and (4) on a parallel computer, there are at least two alternative solution approaches for distributing calculations across processors, an iterative approach and a direct approach. The following characterizes some of the key differences in the two approaches relative to a parallel computer implementation.

Consider, for example, an iterative approach to solving equations (2)-(4). The sequence of calculations in integrating from time stop j to j+1 can be given by equations (5), (6), and (7) as follows:

$$\dot{U}^{r+1}_{j+1} = \dot{U}_j + \frac{k}{2}\left[\ddot{U}_j + \ddot{U}^{r}_{j+1}\right] \tag{5}$$

$$U^{r+1}_{j+1} = U_j + k\dot{U}_j + \frac{k^2}{4}\left[\ddot{U}_j + \ddot{U}^{r}_{j+1}\right] \tag{6}$$

$$M\ddot{U}^{r+1}_{j+1} = -C\dot{U}^{r+1}_{j+1} - KU^{r+1}_{j+1} - G(U^{r+1}_{j+1}) + F(t_{j+1}) \tag{7}$$

where r denotes the rth iteration. If M is diagonal, the equations for each j+1 variable are decoupled during each iteration cycle. If M is not diagonal the decoupling of the equations can be retained by using a Jacobi-type iterative approach to solve equation (7). Convergence can be obtained if the time step is sufficiently small.

A direct approach can also be used to solve equations (2)-(4) for U_{j+1}. For example, equations (2) and (3) can be substituted into equation (4) to obtain

$$\bar{K}U_{j+1} + G(U_{j+1}) = AU_j + B\dot{U}_j + D\ddot{U}_j + F(t_{j+1}) \tag{8}$$

where

$$\bar{K} = \frac{4}{k^2}M + \frac{2}{k}C + K \tag{9}$$

$$A = \frac{4}{k^2}M + \frac{2}{k}C$$

$$B = \frac{4M}{k} + C$$

$$D = M$$

In this case the equations for the variables at j+i are tightly coupled and a solution can be obtained by decomposing the system matrix $\bar{K}$ into upper/lower triangular matrices. Here a major parallel computation step is to map the decomposition across parallel processors. Similar parallel computational issues occur if equations (8) are solved by the Newton Raphson method.

An alternate timewise integration of equations (1) can also be done through integration procedures other than the Newmark method with similar effects. For example consider the explicit central difference approximation for $\ddot{U}$ and $\dot{U}$ at time t_j

$$\ddot{U}_j = \frac{1}{k^2} [U_{j-1} - 2U_j + U_{j+1}] \tag{10}$$

$$\dot{U}_j = \frac{1}{2k} [-U_{j-1} + U_{j+1}]$$

Writing equation (1) at t_j and using the above approximation (10) leads to

$$\bar{K}U_{j+1} + G(U_j) = AU_j + BU_{j-1} + F(t_j) \tag{11}$$

where

$$\bar{K} = \frac{1}{k^2} [M] + \frac{1}{2k} [C]$$

$$A = \frac{2M}{k^2} - K \tag{12}$$

$$B = \frac{1}{2k}C - \frac{1}{k^2}M$$

The solution of equation (11) proceeds forward in time with a single LDL^T decomposition of $\bar{K}$ and require no iterations. Since the $\bar{K}$ contains only the mass and damping matrices it may be sparce and the decomposition is often fairly simple; furthermore, if M and C are diagonal, the decomposition is trivial.

The implementation of the solution to equations (2)-(4) on a parallel computer raises certain issues on the inherent parallelism in the problem and the way in which calculations can be most effectively carried out in parallel. Iteration-type implicit methods such as equations (5-7) or explicit methods such as equations (10,11) lend themselves to assigning equations for a select set of nodal variables to specific processors. The solution procedure has thus been parallelized according to physical regions, and might be denoted "physically parallel". On the other hand, if the solution procedure is based on a Cholesky type decomposition, the solution is parallelized by assigning to each processor various mathematical steps such as matrix algebra. The solution has been parallelized mathematically and might be denoted as "algorithmically parallel". Thus the integration approach, system matrices, number of available processors, and other features may significantly affect the solution approach most appropriate for a problem and the attendent benefits of parallelism.

Each time integration method has certain inherent properties in such areas as accuracy, convergence and stability. Once an integration method has been selected for a crash dynamics analysis certain key steps in the method lend themselves to parallel computations; some of these include

Explicit time integration

Element matrix generation

System matrix assembly

System matrix decomposition

The studies reported herein have focused on testing the parallel benefits of these key steps to help develop an overall crash analysis strategy. The results in these studies are summarized in the following sections.

EXPLICIT TIME INTEGRATION

To investigate explicit time integration methods consider the nonlinear dynamic response of a simply supported shallow arch subjected to a step pulse. A parallel solution to the arch was carried out at Georgia Tech on the FLEX/32 (Fig.1) and similar results were obtained at General Motors on the INTEL iPSC hypercube (Figures 2,3).

The governing equations for the arch are shown in Figure 4 as

$$EI\left(\frac{\partial^4 w}{\partial x^4} - \frac{d^4 w_0}{dx^4}\right) - N\frac{\partial^2 w}{dx^2} + \rho A\frac{\partial^2 w}{dt^2} = -p(x,t) \qquad (13)$$

where

$$N = \frac{EA}{2}\left[\left(\frac{\partial w}{\partial x}\right)^2 - \left(\frac{dw_0}{dx}\right)^2\right]$$

in which:

$w(x,t)$ = transverse displacement of the middle surface of the arch

$w_0(x)$ = initial shape of the arch

ρ = mass density of the arch material

A = cross-sectional area of the arch

p = distributed force per unit length

To solve equations (13) finite differences are used to approximate spacial derivatives and central differences are used for time integration similar to that discussed in equation (4). Since the mass matrix is diagonal the computation sequences were organized as

$$w_i^{iv} = \frac{1}{h^4}\left(w_{i-2} - 4w_{i-1} + 6w_i - 4w_{i+1} + w_{i+2}\right) \qquad (14)$$

$$w_i^{ii} = \frac{1}{h^2}\left(w_{i-1} - 2w_i + w_{i+1}\right) \qquad i = 1, 2, 3, \ldots m$$

$$w_i^{i} = \frac{1}{2h}\left(-w_{i-1} + w_{i+1}\right)$$

where the i subscripts refers to the deflection at the i^{th} node point and superscripts refer to derivatives with respect to x. With equations (14) equations (13) then take the form of equation (10-12) (with C=0 and M diagonal).

$$\ddot{U}_j + KU_j + G(U_j) = F(t_j) \qquad (15)$$

$$U_{j+1} = \ddot{U}_j k^2 + 2U_j - U_{j-1} \qquad (16)$$

Here U_j represents the vector of w; at the j the time step. If m is a multiple of number of the processors n, then equations (15) and (16) can be mapped on a parallel computer with computation distributed equally across processors. If m is not a multiple, some imbalance in processor workload will occur. An executive may be designed in some cases to partition subordinate tasks among processors, help provide a work-leveling approach and minimize idle processors and attendant overhead.

Integration of non-linear structural dynamic equations by the central difference method (equations (15) and (16)) provides a direct approach for calculation the solution at the j+1 time step from information at the j time step. Figures 5, 6 illustrate, for example, how m dynamic equations of motion

are mapped onto m processors (i.e., m=n) for both the FLEX/32 and the INTEL iPSC. In this implementation each processor performs essentially the same task and the integration computations are interrupted once at the end of each time step for communication, as shown in Fig 5,6. Some minor imbalance in computation may occur for equations near the boundary due to minor differences in the equations there. This approach was implemented in concurrent FORTRAN programming language on the FLEX/32 using the shared memory common bus communication logic (Figure 7). It was also implemented at General Motors on the INTEL iPSC hypercube using the interprocessor communication logic (Figure 8).

Figure 9 shows typical results for the nonlinear dynamic response of a specific arch subjected to a step pulse. Figures 10 and 11 show results of computation speedup versus number of processors on the FLEX/32 and Intel iPSC, respectively. Speedup is defined as the time taken to solve a problem on one processor divided by the time to solve the same problem on n processors. Initialization and interprocessor I/O are not included for one processor but are included for n processors. Thus, the speedup on one processor is one, and the theoretical maximum speedup (with no overhead) on n processors is n.

Figure 10 shows that the computation speedup for the FLEX/32 implementation as the number of degrees of freedom (D.O.F.) increases. This result was obtained using the shared memory communication. Similar results for the iPSC are shown in Figure 11. The somewhat better efficiency of the iPSC on the FLEX/32 for the arch problem is due to the local communication nature of the problem which well suited for the iPSC architecture. Similar results for the arch were reported in reference 7 using the locally connected Finite Element Machine (FEM). In reference 7 the speedups approached the ideal values because the communication speed of the FEM was much faster than the processor computation speed. For example, Figure 11 shows that for 120 D.O.F. per

processor a significant improvement in computation speedup occurs and the processor efficiency approaches 100% (Figure 12) for eight processors. On the other hand, with 60 D.O.F. per processor, the efficiency drops off since less computations are required on each processor. Furthermore for 120 D.O.F. the overhead factors such as communication, lack of synchronization, unequal distribution of computation tasks are relatively small. The 99.1% efficiency achieved with eight processors provides a good indication of the potential benefits that can be expected for moderate numbers of processors. The basic results also indicate that the benefits of parallel calculations drop off for those problems which required less extensive calculations and more intensive data communications. A comparison of the FLEX/32 and iPSC results illustrates that a local connection computer such as the hypercube architecture may be more efficient than a shared data base computer like the FLEX/32 for small physically parallel problems such as the arch.

SYSTEM MATRIX DECOMPOSITION

When the time integration procedure uses a direct approach such as that shown in equation (8) a key computationally intensive step is the decomposition of the symmetric system matrix. To investigate the use of parallel processors for this task, a parallel Cholesky decomposition method was developed, implemented on the FLEX/32 and tested on representative matrices with varying band widths. The specific decomposition investigated was

$$K = LDL^T \tag{17}$$

where K is the system matrix, L a lower unit triangular matrix, and D a diagonal matrix

The elements of L and D are given by

$$l_{ij} = \frac{1}{d_{jj}} \left[a_{ij} - \sum_{k=1}^{j-1} d_{kk}\, l_{ik}\, l_{jk} \right] \qquad j<i \tag{18}$$

$$d_{ii} = a_{ii} - \sum_{k=1}^{i-1} d_{kk}\, l_{ik}^2$$

The parallel approach used is based on assigning to each processor the row calculation steps of LD and synchronizing the specific matrix element calculations as required (see for example reference 19).

The results for computation speedup as a function of bandwidth are shown in Figure 13. The results indicated that for half bandwidths above about 50 the speedup with a moderate number of parallel processors is quite good and the average processor idle time is small.

ELEMENT MATRIX GENERATION/ASSEMBLY

A key step in all methods is the generation of element mass and stiffness matrices and the assembly of these matrices into system matrices. The generation task involves many decoupled calculations that are clearly well-suited for parallel computations. The assembly process for the system matrices can be done through row or column assemblies of appropriate element contributions. Some sample investigations were carried out on the FLEX/32 with two different types of elements, one more complex than the other. The generation of element matrices were distributed across available processors. The resulting computation speedups are given in Figure 14 and indicate the expected good performance of a parallel computer in this area. The generation of elements is a very fertile area for benefiting from parallelism in that element computations are largely independent. They do produce large quantities of data and the FLEX/32 shared memory was useful to provide a common data location preparation to assembly.

After the element stiffness matrices were generated, they were assembled into a master system matrix on a row by row basis. The speedup results for the generation and assembly process are given in Figure 14. It is clear from Figure 14 that the generation of elements benefits more from parallelism than does the assembly process. The FLEX/32 shared memory was also especially helpful in managing the large quantity of data associated with the element generation and assembly process.

CONCLUDING REMARKS

A study has been conducted of the potential of parallel computers to improve the computation capabilities for crash analysis of automotive vehicles. The numerical methods have been investigated on test problems with a view toward the steps which are computationally intensive and how these steps lend themselves to parallelism. Key computation steps investigated include matrix generation, assembly and decomposition. Results have also been obtained for an explicit time integration method where the mass matrix is diagonal.

The results indicate that the key computation steps in crash analysis can benefit significantly from parallel computers. They also indicate that there are minimum thresholds in computation tasks for multiple processors to be effective. Below these thresholds data communication becomes a bottleneck and parallel computation efficiency significantly deteriorates. The results suggest that with a relative large number of system degree of freedom and a moderate number of processors, good processor efficiency can be obtained. For example, with one processor per 100 degrees of freedom, parallel computer efficiencies may range from 50% to 90% dependent on the problem. This suggests that with 5000 degree of freedom system and with 50 processors, one might expect computation speedup for effectively coded crash dynamics algorithms to range between 25 and 50, a significant improvement over a sequential computer solution.

These initial results are encouraging and suggest that more detailed studies should be carried out on realistic problems with all major computation steps being implemented in parallel. Two alternate strategies should be considered, one based on an explicit time integration approach which exhibits "physical parallelism" and the second based on an implicit time integration approach which exhibits "algorithmic parallelism."

References

1. Butler, T., and Michel, D. NASTRAN, A Summary of Functions and Capabilities of the NASA Structural Analysis Computer System. NASA SP-260 (1971).

2. Fredriksson, B., and Mackerle, J. Partial List of Major Finite Element Programs and Description of Some of Their Capabilities. In State-of-the-Art Surveys on Finite Element Technology (Noor and Pilkey, Editors). ASME Publication H00290, (1983) pp. 363-403.

3. Noor, A., Storaasli, O., and Fulton, R. Finite Element Technology in the Future. Impact of New Computations on Computational Mechanics. (Noor and Pilkey, Editors), ASME Special Publication H00275, (November 1983) pp. 1-32.

4. Storaasli, O., Peeples, S., Crockett, T., Knott, J., and Adams, L. The Finite Element Machine: An Experiement in Parallel Processing. NASA TM 84514 (1982).

5. Matelan, N., The FLEX/32 Multicomputing Environment. In Research in Structure and Dynamics--1984. NASA CP 2335 (October 1984) pp. 1-14.

6. Storaasli, O., Ransom, J., and Fulton, R. Structural Dynamic Analysis on a Parallel Computer: The Finite Element Machine. AIAA Paper 84-0966-CP, presented at 25th AIAA/ASME/AHS Structures, Structural Dynamics and Materials Conference, Palm Springs, CA (14-16 May 1984).

7. Ransom, J., Storaasli, O., and Fulton R. Application of Concurrent Processing to Structural Dynamic Response Computations. In Research in Structures and Dynamics--1984, NASA CP 2335 (October 1984) pp. 31-44.

8. Bostic, S., and Fulton, R. A Concurrent Processing Approach to Structural Vibration Analysis. 26th AIAA/ASME/ASCE/AHA Structures, Structural Dynamics and Materials Conference, Orlando, FL (15-17 April 1985).

9. Adams, L., and Ortega, J. A Multicolor SOR Method for Parallel Computation. Proc. 1982 Int. Conf. Parallel Processing, pp. 53-56.

10. Adams, L., and Jordon, H. Is SOR Color-Blind? ICASE Report 84-14, NASA-Langley Research Center (1984).

11. Johnson, O., Micchelli, C., and Paul, G. Polynomial Preconditioners for Conjugate Gradient Calculations. SIAM J. Num. Anal. 20 (1983) pp. 362-376.

12. Adams, L. Iterative Algorithms for Large Sparse Linear Systems on Parallel Computers. Ph.D. Thesis, University of Virginia (1982).

13. Adams, L. An M-Step Preconditioned Conjugate Gradient Method for Parallel Computation. Proc. 1983 Int. Conf. Parallel Processing, pp. 36-43.

14. Adams, L. M-Step Preconditioned Conjugate Gradient Methods. To appear in SIAM J. Sci. Stat. Comp.

15. Schreiber, R., and Tang, W. Vectorizing the Conjugate Gradient Method. Proc. Symposium Cyber 200 Applications, Ft. Collins, CO (1982).

16. Poole, E., and Ortega, J. Incomplete Choleski Conjugate Gradient on the Cyber 203/205. Proc. 1984 Cyber 205 Applications Seminar (to appear).

17. Bostic, Susan W., and Fulton, R. Implementation of the Lanczos Method for Structural Vibration Analysis on a Parallel Computer. AIAA Paper No. 86-0930-CP, AIAA/ASME/ASCE/AHS 27th Structures, Structural Dynamics and Materials Conference, San Antonio, TX (19-21 May 1986).

18. Fulton, R. E. The Impact of Parallel Computing on Finite Element Computations. Presented at International Conference on Reliability of Methods for Engineering Analysis, Swansea, U.K.(9-11 July 1986).

19. George, A., Heath, M.T., and Liu, J., Parallel Cholasky Factorization on a Shared-Memory Multiprocessor. Tech Report ORNL-6124, Oak Ridge National Laboratory, March 1985.

20. Newmark, N. A Method of Computation for Structural Dynamics. J. Eng. Mech. Div., ASCE (July 1959) EM3, pp. 67-94.

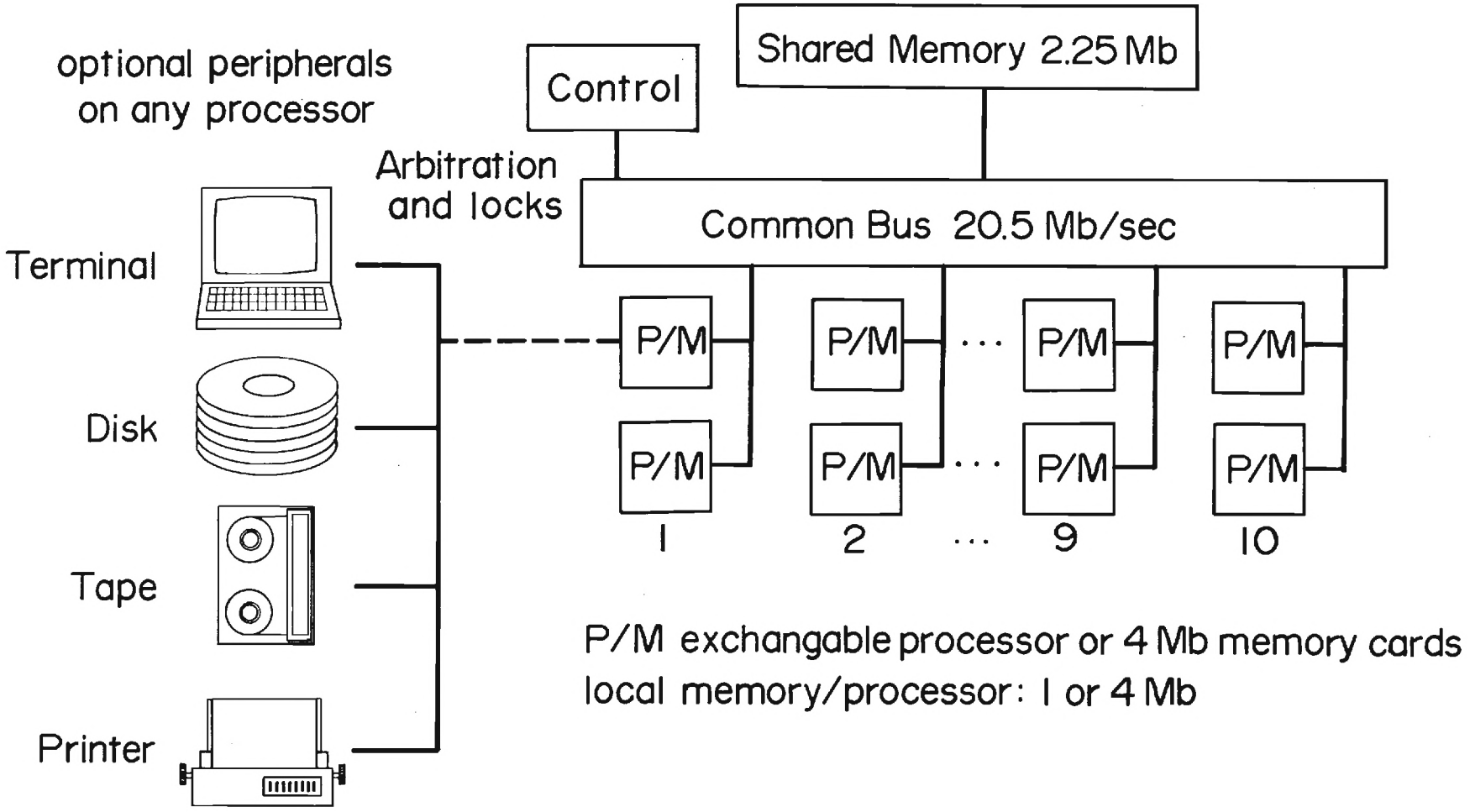
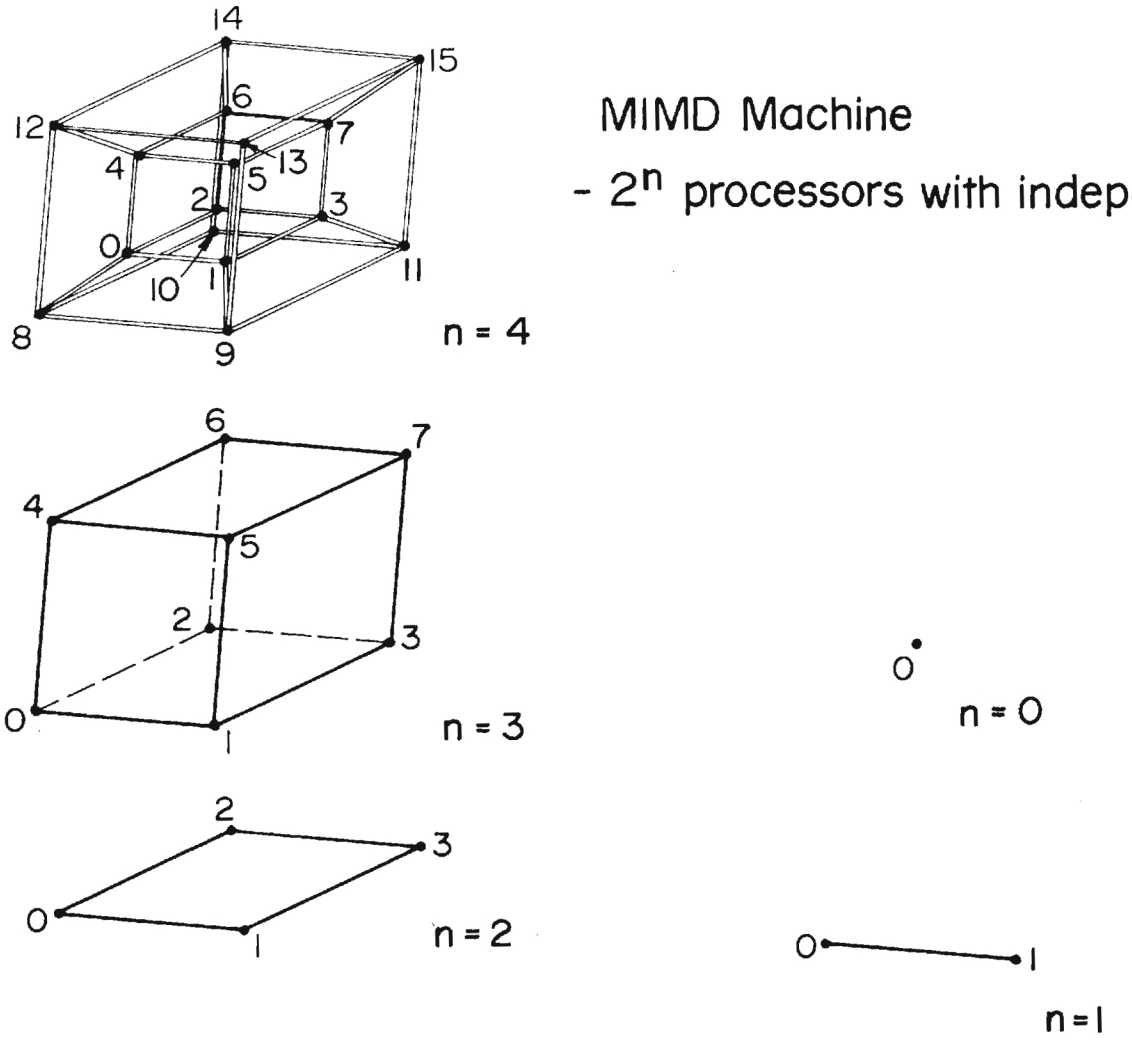


Figure 1. FLEX/32 Multicomputer



MIMD Machine

- 2^n processors with independent memory

Figure 2. iPSC Hypercube Architecture

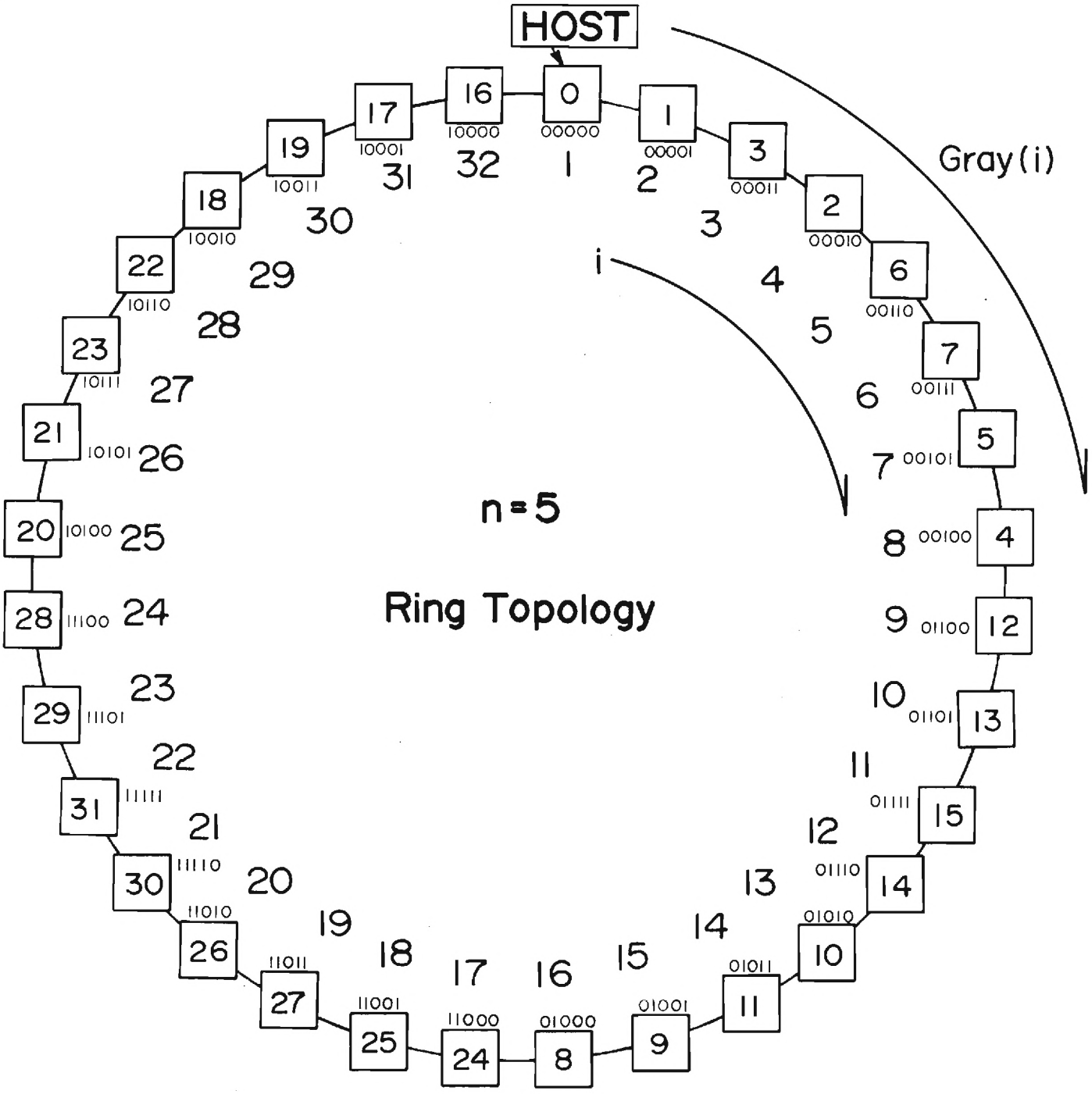


Figure 3. Ring Computation Sequences for Intel iPSC

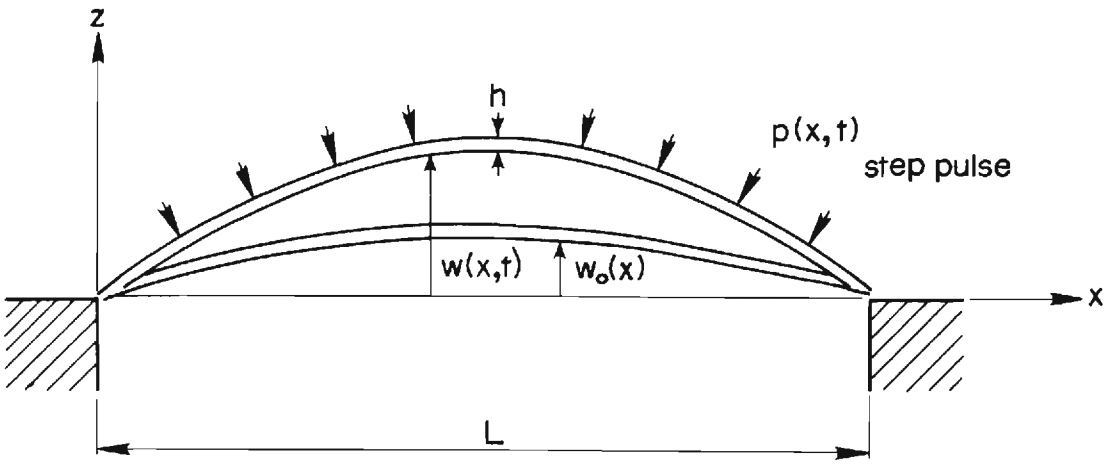


Figure 4. Geometry of a Shallow Arch

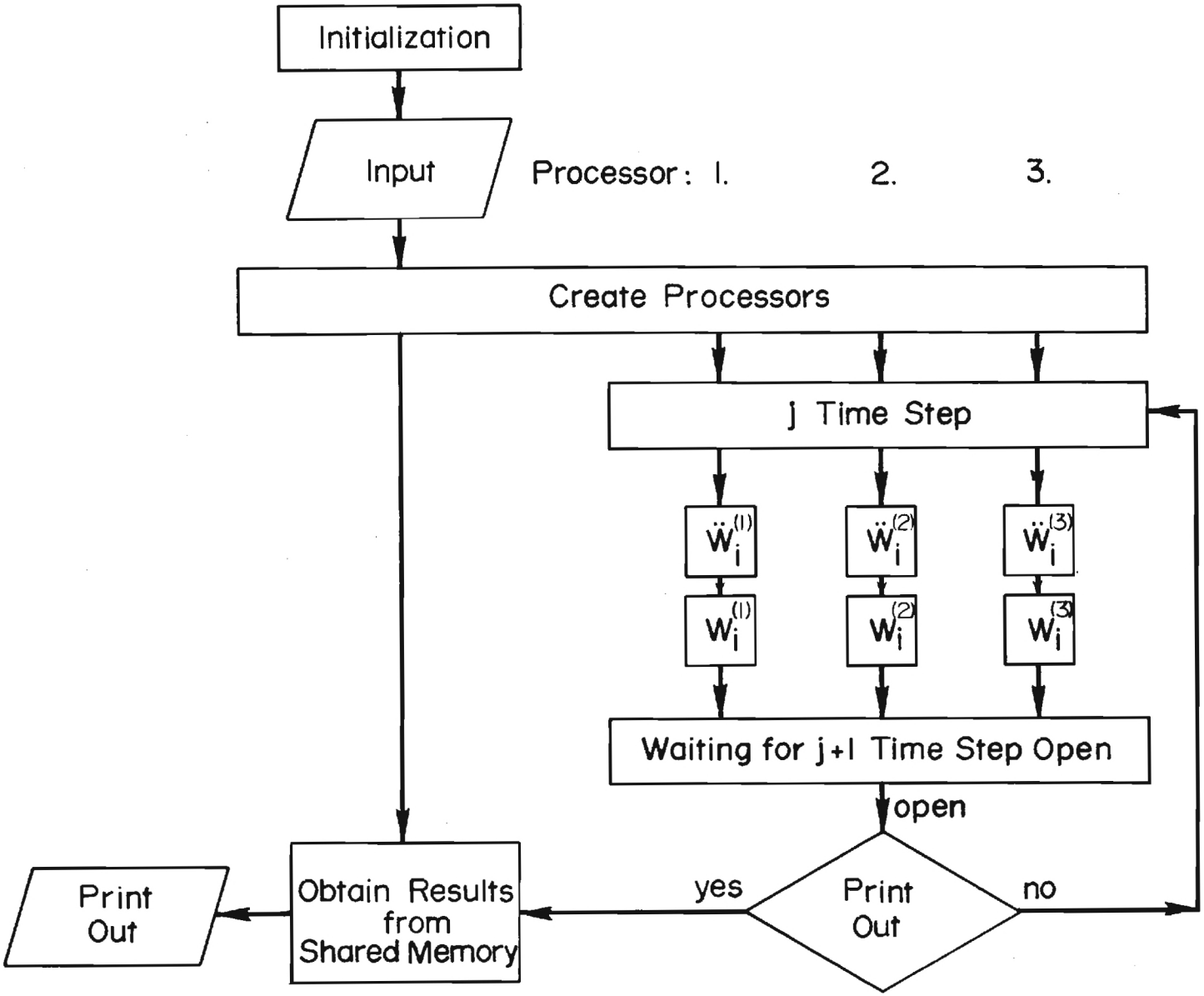


Figure 5. FLEX/32 Integration Approach

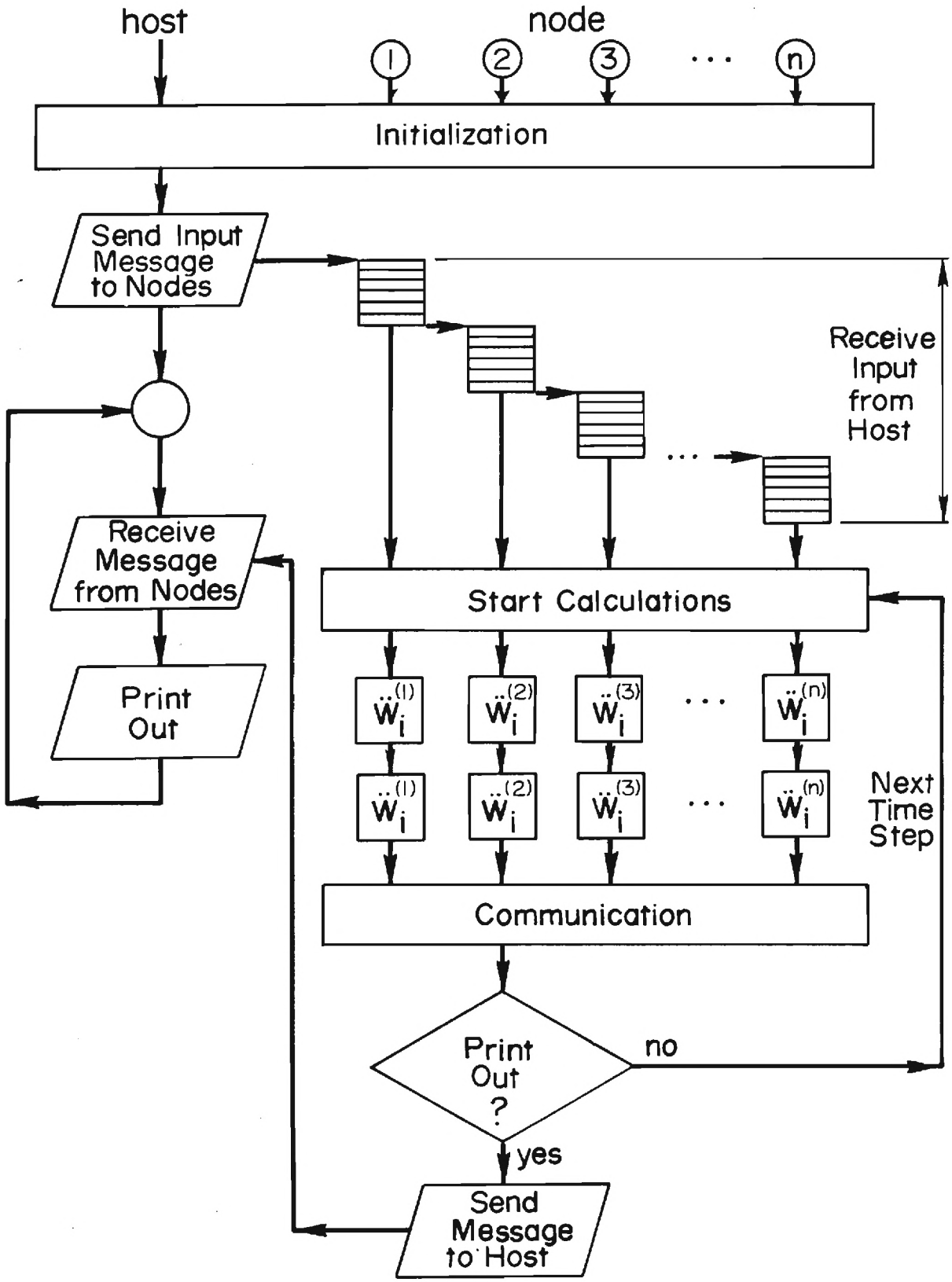


Figure 6. Intel iPSC Integration Approach

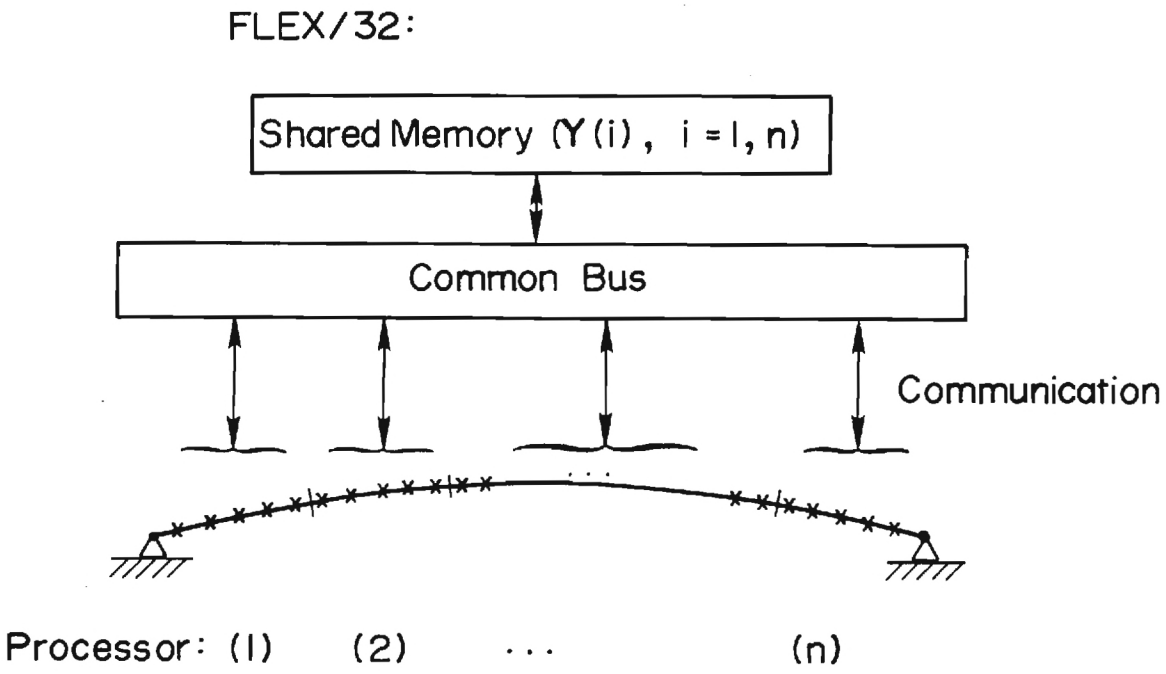


Figure 7. FLEX/32 Communication Sequence

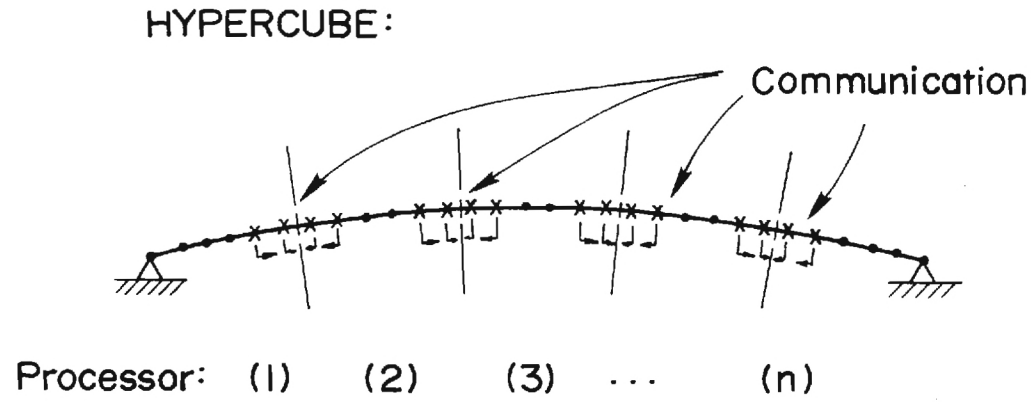


Figure 8. Intel iPSC Communication Sequence

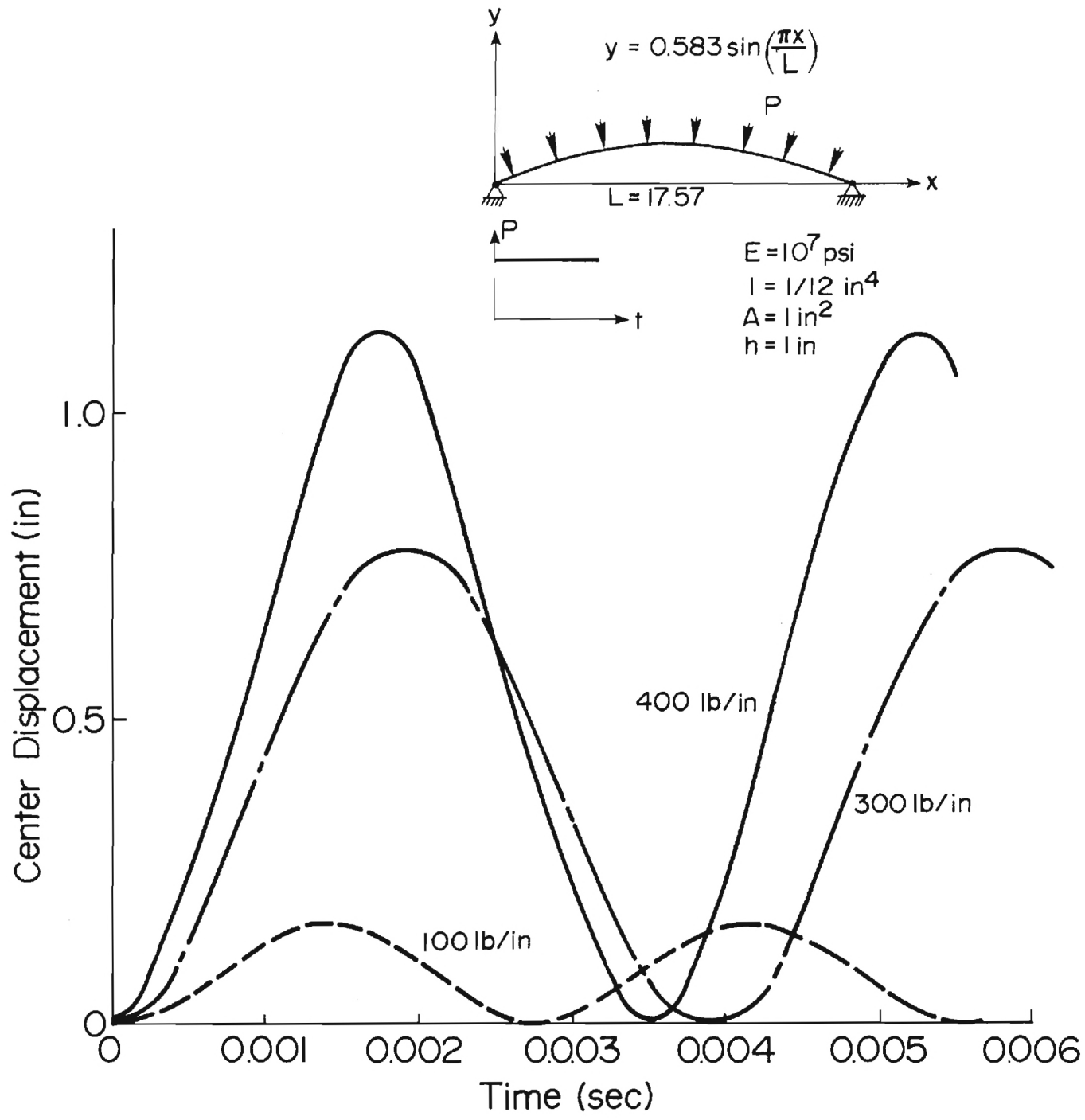


Figure 9. Nonlinear Dynamic Response of Shallow Arch

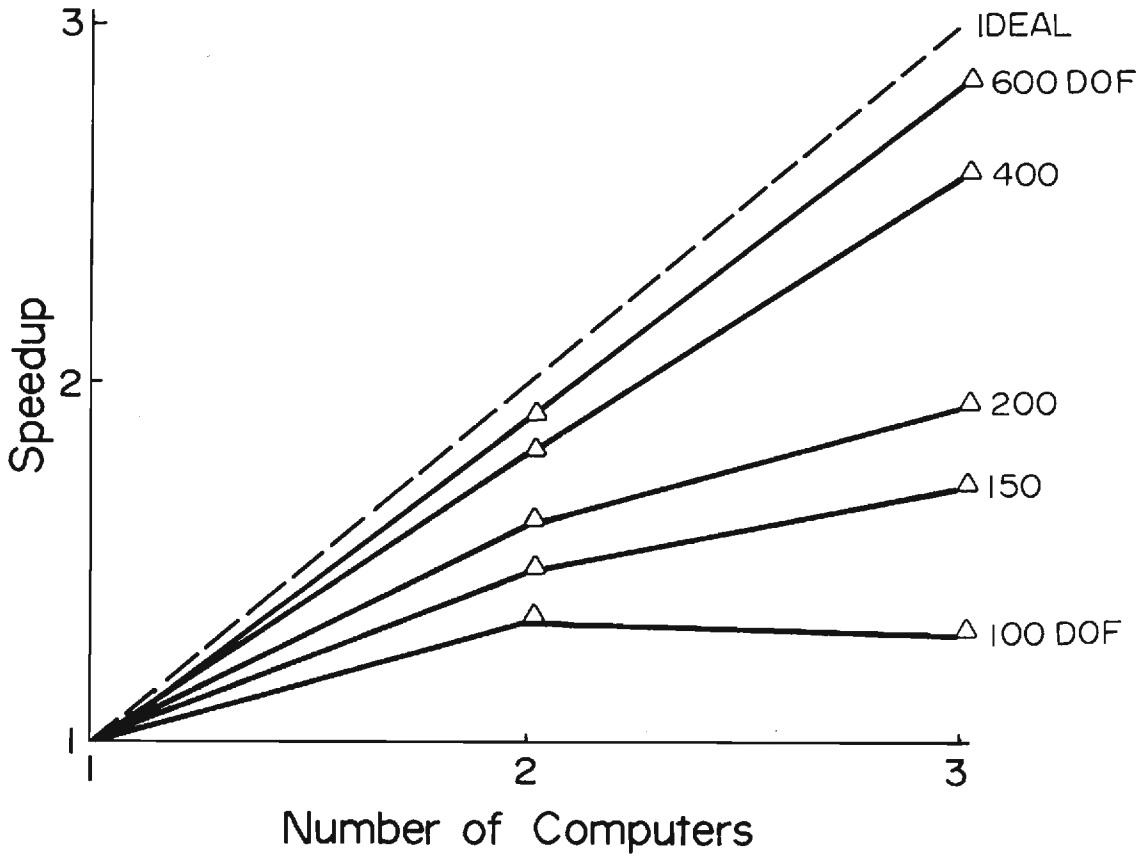


Figure 10. Speedup for Shallow Arch

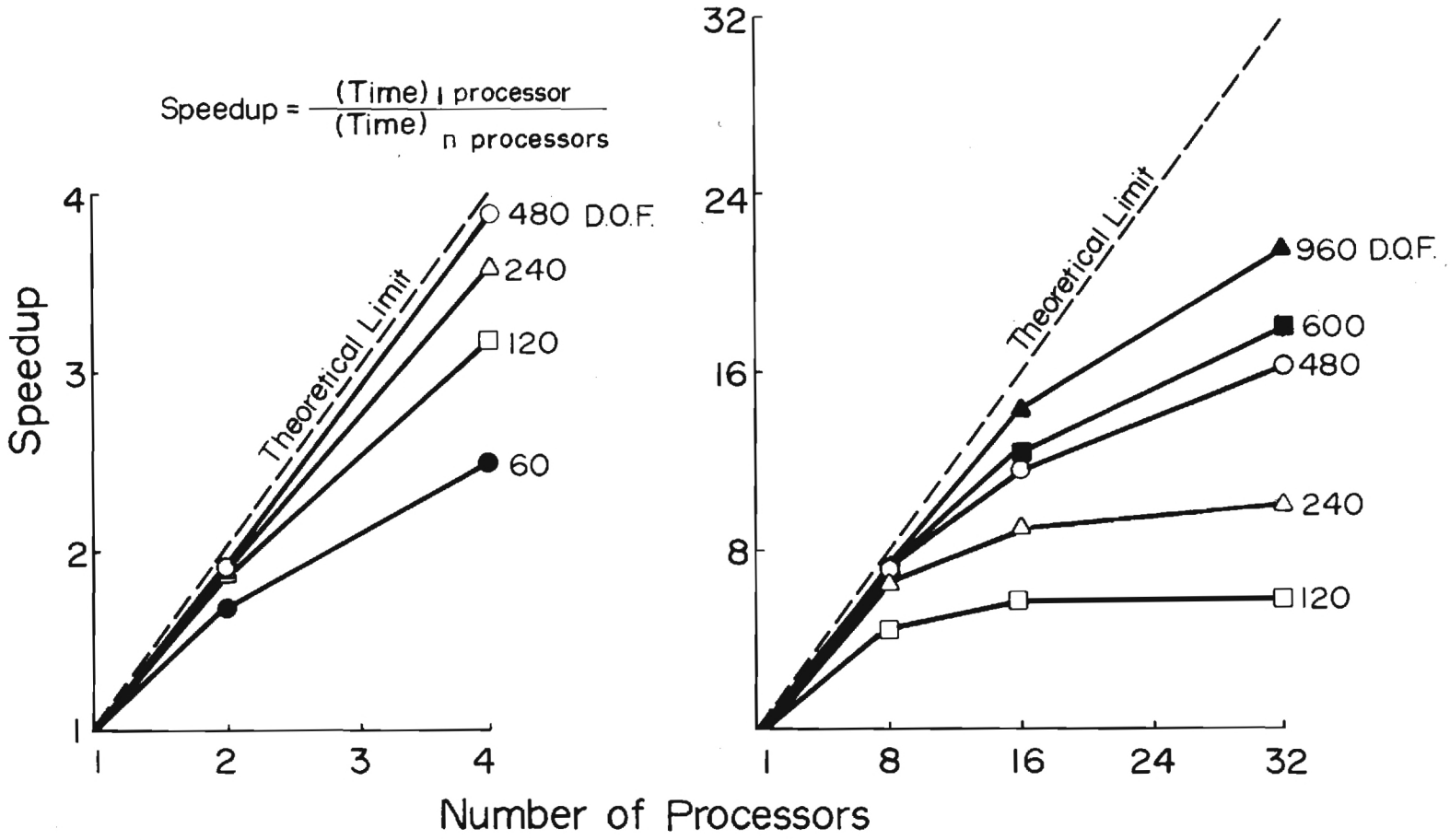


Figure 11. Speedup for Shallow Arch (Intel iPSC)

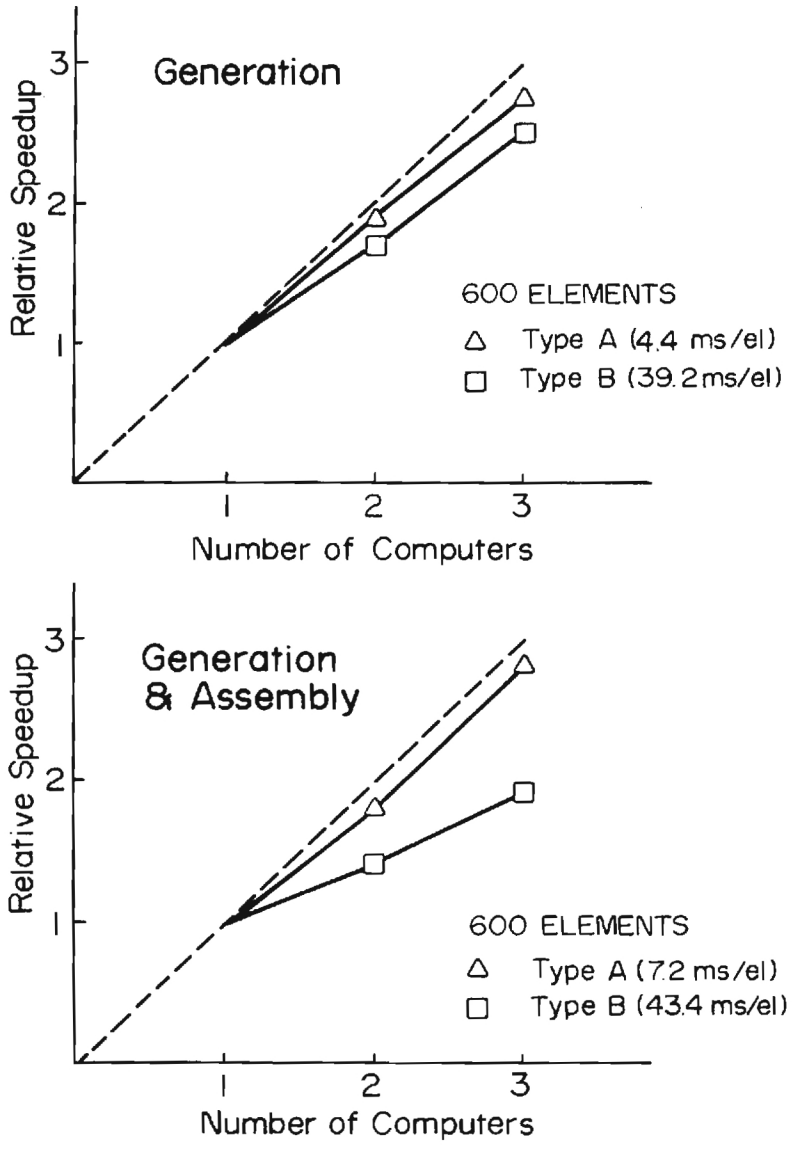


Figure 14. Generation and Assembly of Truss Elements

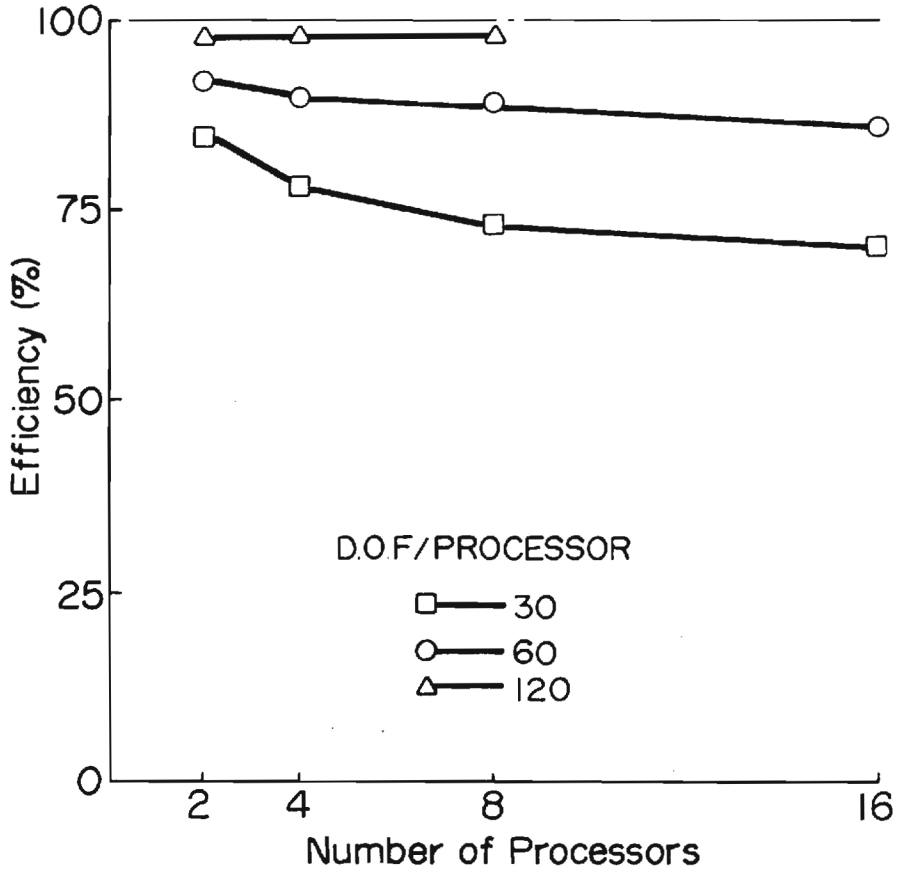
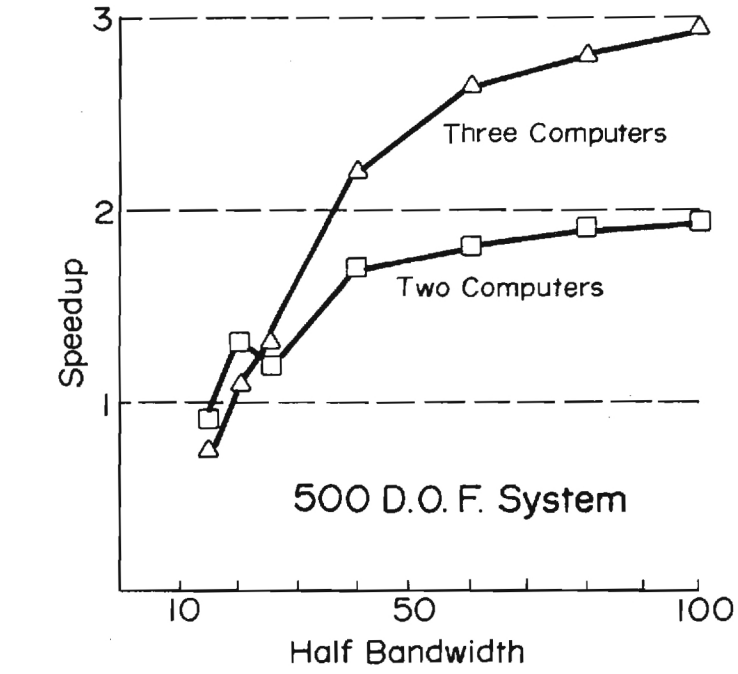


Figure 12. Processor Efficiency (Intel iPSC)



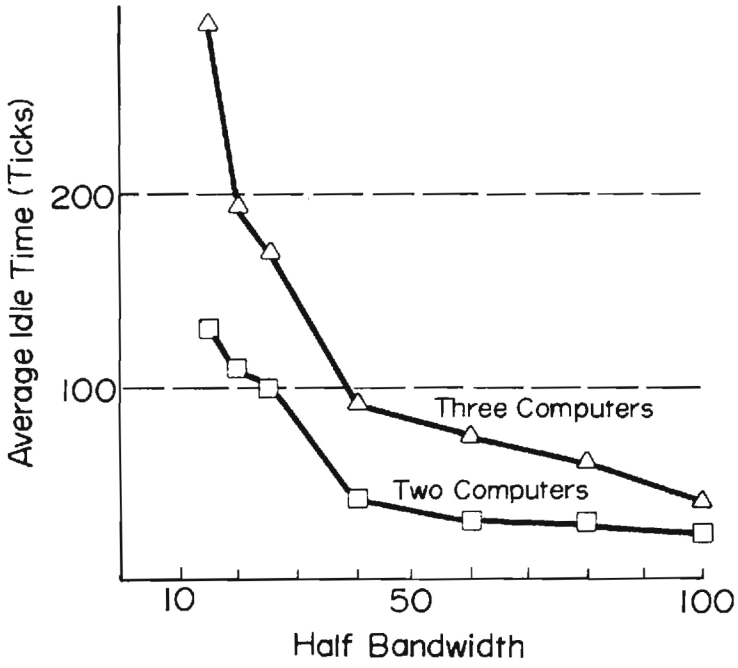


Figure 13. LDLT Decomposition (Banded Matrix)
FLEX/32

COMPUTATIONAL CRASH DYNAMICS METHODS FOR FIFTH GENERATION SUPERCOMPUTERS

COMPUTATIONAL CRASH DYNAMICS METHODS FOR FIFTH GENERATION SUPERCOMPUTERS

February 5th, 1988

Robert E. Fulton
Kuo-Ning Chiang

Georgia Institute of Technology
CAE/CAD Laboratory
Atlanta, GA 30332-0140

Abstract

In the area of crash impact, research is critically needed on the development and evaluation of parallel methods for crash dynamic analysis of complex nonlinear finite element and/or finite difference structure problems. The development of general-purpose finite element structure analysis computer programs have provided the capability to address a wide range of structures problems. However, these software systems are severely limited for nonlinear transient calculations because of the available speed on current sequential computers.

Projected advances in computer technology indicate that significant increases in effective calculation speed will available in the 1990's, through fifth generation supercomputer architectures consisting of arrays of processors operating in parallel on different tasks (see e.g., ref. 3 for a survey). Such advanced supercomputers, denoted as MIMD (multiple instruction, multiple data) computers, have the potential for increasing effective calculation speeds by several orders of magnitude. But this potential increase in speed can not be effectively utilized without the development and implementation of appropriate numerical algorithms for structures which take advantage of the parallel computation features of this new generation of computers. Use of existing conventional algorithms and software will not realize the full potential of these new MIMD computers, and research is needed in the development of parallel structure analysis/design algorithms for these computers (ref. 4-19). This research has been to develop and evaluate parallel methods for crash dynamic analyses of complex nonlinear finite element and finite difference structure problems.

Table of Contents

0. Abstract

1. Introduction 1

2. Literature Survey 3

- Parallel Computers 3
- FLEX/32 and Hypercube iPSC Architecture 3
- Parallel Algorithms 5
- Crash Dynamics 7

3. Proposed Research 9

4. Basic Theory 10

- Explicit Time Integration Scheme 10
- Implicit Time Integration Scheme 11

5. The FENRIS Cholesky Decomposition 15

6. Issues with Shared memory and Hypercube Implementation 17

7. Evaluation of FENRIS/ANSYS for Parallel Implementation 19

- Wave Front and Cholesky Equation Solver 19
- Portability and Mantainability 20
- Evaluation of FENRIS for Parallel Implementation 21

8. Summary 24

- Reference 26
- APPENDIX A - FENRIS 30
- APPENDIX B - ANSYS 35

1. Introduction

Equations of large-order crash dynamic problems are often intractable on current sequential computers due to memory and execution time limitations (ref. 3). In the area of crash impact, research is critically needed on the development and evaluation of suitable parallel methods for nonlinear finite element structural programs. Multiple Instruction Multiple Data (MIMD) computers provides an opportunity for significant gains in computing speed (ref. 21, 24 and 26), and can make tractable the solution to large-order problems.

Recent advances in computer hardware technology have resulted in the introduction of a variety of innovative computer architectures, including vector and parallel processors. The computational power of these new computer systems offers the structural dynamics analyses the opportunity to solve refined models of complex engineering problems and reduce computation times. To exploit this opportunity, concurrent methods for solution to structural dynamic problems need to be investigated. New algorithms, programming rules, and operating systems are needed to utilize the full potential of these new systems. Existing sequential algorithm need to be modified or replaced by algorithms which take advantage of the power of the new systems.

Parallel computers offer the promise for solution to complex problems in detail considered too time-consuming for modern sequential processors. To benefit from advances in parallel computers, software must be developed to encourage and support the migration of programs to this new class of computers. Parallel processing implies two or more processes executing simultaneously on separate computers. Often one process depends on information calculated by other processors. Programming for this environment introduces increased complexity in the software design and algorithmic strategy.

In adapting a solution to parallel processing implementation, one looks for those tasks that can be carried out concurrently with a minimum exchange of information or data dependencies. The algorithm must consider such things as: communication overhead, parallel I/O, memory contention, critical regions of the code that must be executed sequentially, the time to initiate tasks, data interdependency, and idle time resulting from an imbalance of the work load. The implementation reported on this study addresses the above considerations with respect to the solution procedure, portability and maintainability.

Research has been continuing on methods developments and to incorporate a parallel approach into a commercial finite element system. As a result of the study Georgia Tech obtained for the study a reduced ANSYS system from Swanson Analysis and a complete FENRIS system from Veritas Research in Norway. Cooperative agreements have been achieved with both companies so that either ANSYS or FENRIS can be used in the crash dynamics study.

2. Literature Survey

• Parallel Computers

The impact of supercomputers, and highly parallel processing systems on finite element computation is profound. This topic has been reviewed in detail by Noor, Storaasli, and Fulton (ref. 3). Parallel computers can be classified as either SIMD (single instruction multiple data) or MIMD (multiple instruction multiple data) architectures (Figure 3). In the class of SIMD are such machines as the CDC STAR-100, CYBER 203, CYBER 205 and CRAY-1; these provide high computation speed, but are generally not well suited for large-scale finite element computations. Complex structural dynamics problems will require effective computer speeds much greater than 10^3 MFLOPS (million floating point operations per second) for timely results (ref. 3 and Figure 4), and these speeds are well beyond the capabilities of current SIMD computers. A more effective way of increasing computation speeds is through the use of multiprocessor computers (ref. 4 - 19, 59, 60). MIMD computers can be classified by their memory arrangement and the means by which each processor can access data stored in memory. In this classification there are two types of architectures; (1) shared and/or distributed memory architectures such as the CRAY X-MP, CRAY-2, ETA-10, ALLIANT FX/8 and FLEX/32, and (2) network or local memory message based architectures such as the FPS T-series, iPSC INTEL, N-CUBE, AMETEK, and Caltech MARK 111 Hypercube. Representative of shared and local memory architectures are described in the following section.

• The FLEX/32 and iPSC Hypercube Architecture

The FLEX/32 shared memory multicomputer (ref. 61) currently installed at Georgia Tech CAE/CAD Laboratory is composed of 8 processors (can be extended to

20 processors). Each processor has 4 Mbyte of local memory, and all processors can access 2.25 Mbyte of shared memory through a common bus. Figure 1 illustrates the configuration of the 20 processor configuration. Pairs of processors reside on each of 10 local buses,and all 20 machines must access the shared memory by going through the local then the common bus. The processors are numbered as 1, 2, 3, 4, 11, 12, 13, 14 at Georgia Tech. Processor one operates under UNIX operation system and is used for software development and for loading and booting up the remaining processors when a parallel program is to be run. Thus, under normal operation, 7 processors are available for parallel processing and run under the MMOS concurrent operating system. MMOS was designed to fully support multicomputing, multitasking, and real-time programming on the FLEX/32 multicomputer. It is a real-time operating system that contains the facilities (utilities and library routines) to compile, assemble, link/load, and run concurrent and sequential programs.

The Intel iPSC Hypercube multiprocessor (ref. 40) consists of $p = 2^D$ (Figure 2) independent processors or nodes where D is called the dimension of the Hypercube. The machine comes in three models - the D5, D6 and D7. These have respectively 32, 64 and 128 processors. Each processor is a sequential computer processing its own local memory. There is no shared memory available and the processors coorporate by message passing. In addition to the p nodes there must be one additional processor called the host. The host processor should have a relatively large amount of Random Access Memory (RAM) in comparison to the smaller amount of RAM available on the individual hypercube processors. The job of the host is to spawn processes on the nodes, to collect information from them and to handle I/O.

The nodes are identified with the binary representations of the numbers 0 through p-1. Two processors will be called "nearest neighbors" only if the binary representation

of their number differ by one bit. For example, when p = 8 (Figure 2), node number 7 (111) is directly connected to node number 3 (011) or 6 (110) but not to node number 1 (001) or 2 (010). The algorithms discussed in this study have been implemented on an Intel iPSC/d5 machine which is a 32 processors hypercube at the General Motors research Laboratories. The individual processors are the Intel 80286/80287 with up to 512Kb of RAM, The host processor is the Intel 286/310 microcomputer with 2 Mb of RAM and the interconnections are provided by high- speed Ethernets, using the Intel Ethernet chip.

• Parallel Algorithms

In adapting a solution to parallel processing implementation, efficient algorithms typically maximize parallel and minimize sequential calculation. The algorithm must consider such things as: efficient interaction of vector and parallel computations, communication/waiting overhead, memory contention, critical regions of the code that must be executed sequentially, data interdependency, idle time resulting from imbalance of the workload, portability and reliability. In general, for concurrent transient finite element systems two alternate strategies should be considered, one based on an explicit method and the other based on an implicit method. An obvious way to carry out a concurrent execution of explicit finite element method using n processors is to decompose the structural domain into n regions and give each processor "responsibility" for solving an entire subproblem. A substructuring method for implementation on multiprocessor computers has been recently proposed by Storaasli and Bergan (ref. 31). An automated mesh decomposition and concurrent finite element analysis has been discussed by Malone (ref. 26). His approach used a central difference time integration scheme and selected test problems were implemented on Hypercube multiprocessor computers. The formulation includes a new decomposition algorithm which automatically divides an arbitrary finite element mesh into region and assigns each region to a processor on the Hypercube, The algorithm has been implemented on a 32 processor

Hypercube for plane-stress analysis and exhibits more than 95% efficient use of machine. The Newmark trapezoidal integration approach has been carried out by Storaasli, Ransom and Fulton (ref. 6), and their results for a two dimensional beam grillage problem gave computation speedups reaching a value of 6.5 for eight processors.

When the time integration procedure uses a direct approach, a key computationally intensive step is the decomposition of the symmetric/unsymmetric system matrix. The Cholesky decomposition (a scheme widely used in the existing finite element equation solver) has been studied by Goehlich, Komzsik and Fulton (ref. 21). Their parallel approach was incorporated in MSC/Nastran and tested for several static stress analysis demostration problems on CRAY X-MP and IBM 3090 computers with up to four processors. The results indicated that a parallel processing approach can significantly reduce execution time for large scale finite element problems. A parallel Cholesky scheme is also discussed by Bostic and Fulton (ref. 58) in their Lanczos method implementations. In the Lanczos method the decomposition step is the most time-consuming calculation step for large problems and this step benefited the most from the parallel procedure. A speedup of 7.8 on eight processors was obtained for the decomposition step of the mast problem on the FLEX/32 Multicomputer.

It is well known that local memory machines have difficulty in achieving a good speedup for algorithms such as the Cholesky decomposition. The sequential Frontal method was first described by Irons (ref. 37) to solve finite element problems, and is based on the observation that Gaussian elimination can begin before assembly of the K matrix is completed. In fact a node in the finite element mesh can be eliminated as soon as all the elements associated with it are assembled. A parallel multifrontal equation solver was investigated by Geist (ref. 22) using a local memory multiprocessor (iPSC Hypercube). The approach is to have one front on each processor and is designed so that

each processor solves an entire subproblem. While the limited results are encouraging it requires regeneration of the K matrix in every iteration, a strategy not well suited for a general purpose nonlinear finite element analysis.

An alternative approach is the conjugate gradient iteration equation solver, a method studied by Lyzenga, et al (ref. 29), and Nour-Omid and Parks (ref. 56) on a 32 processor Caltech Hypercube. It was used successfully for a large number of finite element problems. The conjugate gradient algorithm is well-suited for vector computation but, because of its many synchronization points, it more difficult for parallel computation. A block conjugate gradient algorithm has been discussed by O'Leary (ref. 53), and the parallel efficiency of his algorithm exceeded that of the standard conjugate gradient algorithm. The block preconditioned conjugate gradient method has also been investigated by Meurant (ref. 56) on a four processors computer CRAY X-MP/48, where the numerical results exhibit good parallel and vector efficiency. Other studies of parallel equation solvers have focused on narrow banded systems using such methods as cyclic reduction, recusive doubling, divide and conquer and twisted factorization methods (ref. 42, 54 and 55). The results show a good speedup potential in both parallel and vector computations.

• Crash Dynamics

The associated tasks for crashworthiness are: prediction of the crashworthiness of a specific structure, improvement in design, and possibly optimization of the structure with respect to crashworthiness. In the past, the evaluation of crash performance was dependant on the impact tests, but such tests are costly and require a long preparation time (ref. 23, 39). Therefore, numerical studies of crash analyses are becoming increasingly important in this area. For a complicated crash simulation, the amount of data required to describe the problem is enormous, and results in extremely large computational costs. One of the first finite element programs to take full advantage of vector

processes, such as the CRAY-1, is DYNA3D (ref. 52, 57). The DYNA3D system has been reported by Benson, Hallquist and Stillman (ref. 57). One of the test problem shown at Figure 5, is a curved S-Frame impacted by a large mass at 30 km/hour. The algorithm used for the dynamic analysis was the explicit central difference method. The finite element model consists of 1600 shell elements with five integration points through the thickness. The computation for 35 ms of real time required nearly 4.2 CPU hours on the CRAY-1.

Computer simulation of crash phenomena has also been studied by Argyris, Balmer and Kurz (ref. 30) using the S-Frame test problem (see Figure 6). Their finite element model of the S-Frame is comprosed of 388 TRUMP element with 1200 unknowns, and the dynamic computation was performed via a matrix solution of Newmark's method (ref. 20). The time increment was 0.0005 sec and the number of iterations per time step was limited to 5. Computation up to 45 ms (90 time intervals) required 1.5 CPU hours on the CRAY-1M (the estimated CPU-time on VAX-11/780 was 300 hours).

The results obtained do not adequately fit the experimental result, and a refined mesh and smaller time step is required to obtain better results. The results discussed here show that todays supercomputers are marginal at best for sophisticated crashworthiness calculation. This implies that the solution algorithm must not only be inherently efficient, but also that it must take advantage of both vector and parallel computers so that all of a supercomputer's potential is realized. In the area of crash impact, research is critically need on the development and evaluation of vector and parallel methods for crash dynamic analysis of complex nonlinear transient finite element problems.

3. Proposed Research

The research reported herein is to investigate the potential of parallel computers to significantly improve the capability for crash analysis of automotive vehicles. Task planned are:

(1) the development of concurrent algorithms for nonlinear transient analysis which make effective use of multiprocessor computing power. These algorithms include (1) implicit methods such as Newmark type which build on Cholesky decomposition strategy and exhibits "algorithmic parallelism", and (2) explicit method such as the central difference time integration scheme which exhibits "physical parallelism"

(2) An assessment of the potential of the shared memory and local memory computers for use in parallel computations

(3) Implement the FENRIS nonlinear finite element commercial program on the FLEX/32 MMOS (Multicomputing Multitasking Operating System) for parallel solution

(4) Incorporate a parallel Cholesky decomposition method in FENRIS and evaluate the performance of the resulting parallel nonlinear finite element system

(5) Investigate the implementation and performance of a parallel force vector generation in FENRIS.

(6) Evaluation the methods with the GM S-Frame test problem.

Work has progressed well toward the research goals, some specific comments relative to the above six goals will be discussed on the following sections.

4. Basic Theory

There are two basic integration approaches used in the transient finite element systems: (1) explicit method and (2) implicit method. In an implicit formulation, the temporal difference equations such as velocities and accelerations are combined with the equation of motion so that the displacements at a new time step can be calculated directly. For nonlinear problems, however, some iteration approaches like the Newton Raphson method must also be used. In an explicit formulation the response quantities at the new time step are expressed in terms of previously determinated values of displacement, velocity and require no iterations.

• Explicit Time Integration Scheme

In general the finite element equations governing the dynamic response of a structure may take the form

$$R_{int} = R^E - R^D - R^I \tag{1}$$

where R^E, R_{int}, R^D and R^I denote external forces, internal forces, damping force, inertia force and can be expressed as

$$\begin{aligned} R^D &= C\,\dot{u} \\ R_{int} &= K\,\Delta u \\ R^I &= M\,\ddot{u} \end{aligned} \tag{2}$$

where C is the damping matrix (Appendix A), K is the stiffness matrix and M is the mass matrix.

The explicit central difference approximation for u and $\dot{u}$ from time step j to j+1 over the time interval h is

$$\begin{aligned} \dot{u}_{j+1/2} &= \dot{u}_{j-1/2} + h\,\ddot{u}_j \\ \dot{u}_{j+1} &= \dot{u}_{j-1} + 2\,h\,\ddot{u}_j \\ u_{j+1} &= u_j + h\,\dot{u}_{j+1/2} \end{aligned} \tag{3}$$

Rewriting equation (1) and using the above approximations leads to

$$M\ddot{u}_{j+1} = R^E_{j+1} - C\dot{u}_{j+1} - K\Delta u_{j+1} \tag{4}$$

For both explicit or implicit time integration schemes, the residual force $\hat{Q}$ vector must vanish; thus

$$\begin{aligned} \hat{Q} &= R^E_{j+1} - M\ddot{u}_{j+1} - C\dot{u}_{j+1} - K\Delta u_{j+1} \\ &= 0 \end{aligned} \tag{5}$$

The explicit time integration scheme automatically satisfies equation (5) and require no iteration; if M is lumped, the decomposition is trivial.

Parallel flow chart for explicit finite element program is shown at Figure 7. Each processor computes all assigned element quantities such as C_{sub}, K_{sub} matrix, stresses_{sub}, strains_{sub} and $\text{accelerations}_{sub}$ without data exchanging and these quantities are stored in the local memory. However, the lumped mass matrix, nonlinear force vectors $\hat{F}_{sys}$, displacement and velocity vectors are put into global memory for use by more than one processor. From transient finite element system points of view, parallel I/O, less memory contention and high parallel computation efficiency make explicit methods well suited for parallel computations.

• Implicit Time Integration Scheme

In this algorithm the discussion will focus on parallel matrix decomposition, and

how to use local memory as a secondary memory storage. A general incremental-iterative scheme combine with the Newton Raphson family and Newmark method are used for solving the dynamic equations and the final equation can be expressed as (more detail can see Appendix A)

$$\hat{K}\Delta\hat{u}_{i+1} = \Delta\hat{R}_{i+1} \tag{6}$$

The effective stiffness matrix $\hat{K}$ and the incremental residual vector $\Delta\hat{R}_{i+1}$ are

$$\hat{K} = a_o M + c_o C_V + K_T$$
$$\Delta\hat{R}_{i+1} = R^E_{i+1} - R_{int,i} + M[\hat{a}_i + \hat{b}_i(\alpha_1 - a_o\alpha_2) - \ddot{u}_i] + C_V\hat{b}_i(1 - c_o\alpha_2)$$

The corresponding increments of the velocities and accelerations are

$$\Delta\dot{u}_{i+1} = f_1(\Delta u_{i+1}, \dot{u}_i, \ddot{u}_i) \tag{7}$$

$$\Delta\ddot{u}_{i+1} = f_2(\Delta u_{i+1}, \dot{u}_i, \ddot{u}_i) \tag{8}$$

If the vectors $u_{i+1}, \dot{u}_{i+1}$ and $\ddot{u}_{i+1}$ are substituted into the dynamic equilibrium equation (1), it will be found for nonlinear cases that this equation is not completely satisfied, and equilibrium iterations must be carried out.

In the implicit time integration scheme the nonlinear force vectors are generated by "physical/substructural parallelism"; however, the Cholesky decomposition is solved by "algorithmic parallelism". The major time-comsuming steps of this algorithm are (1) nonlinear force vectors generation, (2) input/output, and (3) matrix decomposition. The mass matrix, stiffness matrix, viscous damping matrix, internal force vectors can be computed element by element and assembled into the global matrices M, K,

C_V and R_{int}. These computational tasks are independent and highly parallelizable and can be assigned across processors. The Cholesky decomposition step is the most time-comsuming calculation for large size finite element problems, and this method has shown good speedups through parallel and vector processing, especially when the half-bandwidth is sufficiently large (ref. 21, 25).

For a large structural systems the effective stiffness matrix cannot be accommodated in core storage (shared memory), and must be segmented into blocks (ref. 27). These blocks must then be stored temporary on low speed auxiliary storage, usually as disk files, where the CPU time is relatively high. Shared/Local memory can take advantage of the local memory available on each processor, Data blocks are stored in local core memory by use of standard FORTRAN 77 calculations such as Local Variables 1 to n equal to Share Variables 1 to n, these data mapping CPU times are much less than the standard I/O operation. Therefore, the use of local memory as a secondary storage unit is needed for parallel finite element implementations.

The equation (6) may take the form as A X = B, where the effective stiffness matrix A = L U = L D^{-1} L^T. Here the matrix L is a lower unit triangular matrix and D is a diagonal matrix, the decomposition is carried out in three step:

$$
\begin{aligned}
l_{ij} &= a_{ij} - \sum_{n=1}^{i-1} l_{ni}\, l_{nj} \qquad i < j \\
l_{ij} &= l_{ij}\; d_{ii}^{-1} \\
d_{jj}^{-1} &= 1 \,/\, (a_{jj} - \sum_{n=1}^{j-1} l_{nj}\, l_{nj}\, d_{nn}) \qquad \text{for all j}
\end{aligned}
\tag{9}
$$

The matrix L has the same shape as A (skyline form) and is therefore stored in the same location. The diagonal matrix D^{-1} is stored in an array residing in core. The stiffness

matrix A is divided into several blocks, each consisting of several columns and A is triangulized block by block. Some typical results of this algorithm are shown on the following section.

5. The FENRIS Cholesky Decomposition

Work has progress well toward the research goals. A significant effort has been completed to evaluate, select and obtain a software testbed for parallel implementation of crash dynamics analyses. Criteria for selecting the system include the following: commercial quality finite element system, nonlinear material and geometry, dynamic loads, FORTRAN, suitably documented and available source code. A number of candidate systems were investigated including NASTRAN, ANSYS, ADINA, FENRIS, DYNA-3D, DYCAST, and SPAR. The availability of both the technical capability and also the source code was difficult to achieve. As a result of the study Georgia Tech obtained a reduced ANSYS system from Swanson Analysis and a complete FENRIS system from Veritas Research in NORWAY. Cooperative agreements have been achieved with both companies so that either ANSYS or FENRIS can be used in the crash dynamics study.

Research has continued on methods development and a parallel FENRIS LDL^T method was developed. Speedups of this parallel Cholesky decomposition (compare with the sequential FENRIS Cholesky Decomposition) are very encouraging. The improvements are achieved by refined parallel computation strategies, and a highly machine independent waiting control routines were used which reduce wait time in the matrix decomposition. A selected static test problem is shown at Figure 8. which is a cantilever beam subjected to a concentrated force. This finite element model comprises 100 quadrilateral elements with 444 unknowns and the maximum half-bandwidth equal to 48. Typical results showing the improved computation speedups are given in Figure 9. The results show computation speedups of up to 6.54 for seven processors and indicate that significant speedups can be achieved through the use of many processors for a appropriate finite element problem. It should be noted that the Cholesky decomposition

is also well-suited for vector computation (Figure 10). In the parallel Cholesky decomposition, each processor is responsible for decomposing the whole column of the system matrix; therefore no vector length penalty is introduced. From above observations the LDL^T algorithm shows a good promise for parallel/vector computation and this algorithm will be the baseline method for the crash dynamic studies.

6. Issues with the Shared Memory and Hypercube Implementations

These studies reported herein investigate the use of the FLEX/32 shared memory and the iPSC Hypercube architectures for similar structural calculations; some observations on the relative merits are useful. As discussed earlier the implementations illustrate how both shared and local memory machines can give good parallel efficiency on problems which have physical characteristics oriented to parallel computations. In the local memory machine, however the communication distance between processors exchanging data is determined by the choice of processor mapping. Thus the major mapping issues are (1) to make data passing between processors possible through a shortest communication path, and (2) to minimize the number of other processors with which each processor must communicate. For the shallow arch test problem this mapping is very easy since the processors can be arranged from the left hand boundary (Figure 11) to the right one. In such a case each interprocessor is connected by a direct communication channel and the last processor does not have to communicate with the first one. However, for some structures such as closed arch or ring problems subjected to unsymmectrical loads, this processors mapping topology would have problems since the last processor must communicate with the first processor. For the iPSC Hypercube the data passing distance for the ring would require twice the time needed for the shallow arch. Optimizing processors mapping and data passing length in a local memory machine is therefore a highly machine dependent task and different local memory machines have different communication efficiency; thus software portability and reliability are major issues for local memory machines.

For shared memory machines one can take advantage of global memory where any processor can access this memory. Communication subroutine can be written in a high level language or transferred to assembler language for better efficiency. Some approach

such as a shared lock flag must be established to provide controlled access to the shared memory. For example, if the lock flag is on, the parallel task is placed in a waiting mode with no data exchange. When the lock flag is off, the parallel task is removed from waiting and global data can be accessed by each processor. Since the flag concept can be coded by high level language, it can be made machine independent. Thus parallel speedups will be consistent across different shared memory computers and there is little difficulty in migrating code to different shared memory computers, such as from the FLEX/32 to the CRAY X-MP/48.

A second strategy is associated with "algorithmic parallelism" such as in the case of the Cholesky decomposition (a scheme widely used in the existing finite element equation solver). Here the matrix diagonal terms are required by each processor for calculating L^T. In the local memory machines these diagonal terms have to be broadcasted to each processor, which results in a sequential communication task. Since the data processing time is relatively large the algorithm did not show a good parallel efficiency on local memory machine. Another disadvantage of local memory machines for parallel finite element implementations is that the system matrices such as K, M, C are generated in blocks ("substructure"), whereas the parallel decomposition algorithm needs the information assigned to processors columnwise. Combination of these two computational steps in a local memory machine is a problem, whereas it is more easily dealt with for shared memory machines because the shared data base is readily available for all processing tasks.

7. Evaluation of FENRIS/ANSYS for Parallel Implementation

Figure 12, 13 show the solution flow charts of ANSYS/FENRIS for the Nonlinear Transient Dynamic Analysis. The major solution procedures include element matrix generation/assembly, effective stiffness matrix ($\hat{K}$) triangulization and nonlinear load vector ($\hat{R}$) calculation. The matrix generation and the nonlinear load vector calculation tasks involve many decoupled calculations that are clearly well suited for parallel computations on both codes (APPENDIX A and B). For nonlinear transient dynamics problems the ANSYS performs a wave front solution stategy, and the system matrix will be regenerated at every iteration. The equation solver adopted by FENRIS is the Cholesky decomposition, and there are four iterations stategies which can be used (see Figure 14). These are:

(1) **Incrementation with unbalanced force correction:** no iteration will be performed within each incremental step and the effective stiffness matrix $\hat{K}$ will be decomposed at every steps.

(2) **Initial stiffness iteration:** $\hat{K}$ matrix will not be updated and only decomposed once with the other steps using the backward/forward substitutions, this method is not well suited for parallel computations since the backward/forward substitutions is not efficiently carried out in parallel.

(3) **Modified Newton-Raphson:** $\hat{K}$ matrix will be updated and decomposed at each new incremental step.

(4) **True Newton-Raphson:** $\hat{K}$ matrix will be updated and decomposed at every iterations.

Two of these four methods, (1) Incrementation with unbalanced force correction and (4) true Newton Raphson show better potential for parallel implementations, however, the first method is not suitable for highly nonlinear problems.

- **Wave Frontal and Cholesky Equation Solvers**

The ANSYS program uses the wave front or frontal equation solver first discussed

by Irons (ref.37) for the system of the simultaneous linear equations associated into the finite element method, the method is based on the observation that Gaussian elimination can begin before the assembly of effective stiffness matrix is completed. The number of equations which are active after any element has been processed during the solution procedure is called the wave front at that point. In this method calculations and I/O are tied together and the front width has to be kept as small as possible in order for frontal method to be efficient. Elements are sequentially added to the effective stiffness matrix resulting in a poor speedup for a parallel version of this algorithm.

The p-frontal (multifrontal) method has been investigated by Geist (ref.22) on the Intel Hypercube. The p-frontal approach is designed so that each processor solves an entire subproblem, The testbed routines read the reorganizing information from data file set up by the engineer. The automation of the partitioning and the reordering of the problem is still under investigation. However, it is not easy to incorporate a parallel approach into ANSYS, because several pre-tasks have to be done on each processor before p-front decomposition. Some of these include new node renumbering routine, sub-matrix (stiffness, mass, damping) generation, sub-nonlinear load vectors calculation, parallel I/O and data rearrange. In contrast, a parallel Cholesky decomposition can be tested independently in a program like FENRIS and does not require the reconstruction of the rest of the software. The speedup of the Cholesky method is good for seven processors when the half-bandwidth greater than 50 (ref.21, 25 and Figure 9). In light of the above discussion a Cholesky based finite element system such as FENRIS appears well suited for parallel implementations.

• Portability and Maintainability

A key issue for a successful finite element system is its availability, portability, and maintainability on different computers. The major advantages of FENRIS are its por-

tability, building block concept, programming and documentation standards. It should be noted that Georgia Tech migrated the full FENRIS VAX Version to FLEX/32 UNIX and MMOS operating system without much difficulty. The building block concept adopted in FENRIS enables a person who is working on a special topic, e.g. on a new solution algorithm such as the Parallel Cholesky Decomposition or on a new constituitive law for a material, this is allow him to program and test his work without having to develop a whole new program. This possibility is especially important for a joint project like a Parallel FENRIS.

Swanson Analysis, Inc. has recently improved its ANSYS User's Manual and Theoretical Manual, and enlarged its well-organized Verification Manual to more than 164 problems. These significant efforts are very helpful for the users who want to know how to use the ANSYS to model and solving their problems. However, for parallel implementations such as recoding or adding new subroutines (e.g. multifrontal algorithm) into ANSYS, more documentations are required than that provided in the source codes and manuals.

Both ANSYS and FENRIS are good commercial finite element system and have their own fans, but from the viewpoint of portability, maintainablilty, code documentation and algorithms the FENRIS appeared to have more promise as a testbed for the parallel implementations.

• Evaluation of FENRIS Performance for Parallel Implementation

The FENRIS system has been initially installed at Georgia Tech on VAX/750 VMS, FLEX/32 UNIX and MMOS operating systems; and initial results obtained for the S-Frame test problem have been achieved on both machines. The test finite element model comprises 180 triangular shell elements with 456 unknowns. The impact phe-

nomenon was computed on the basis of elastic-perfectly plastic constitutive material (A-36 Steel). Dynamic computations were performed via a matrix solution of Newmark's unconditional stable scheme, with a time increment equaled to 0.0005 second. A fine mesh or smaller time increment was not used because the limitation of CPU time and disk space on the VAX/750 and FLEX/32. Typical results are shown as Figure 15, 16; these results compare with favorably with similar results by Argyris, et al, (ref. 30) and indicate that FENRIS system provides an adequate basis for the parallel crash dynamics investigation.

The computation times attributed to both I/O and CPU for the S-Frame test problem are shown at Figure 17. This Figure illustrates that the percentage of computation associated with effective stiffness matrix, internal and external force vectors generation is 84.6 % whereas for equation solving is 11.90 %. However, for a larger models the percentage for equation solving will increase rapidly and it is expected that the equation solving will become the dominant time consuming phase for very large problem with large bandwidths. It is a well known that the Cholesky decomposition has difficulty obtaining an attractive speed up for a small half bandwidth system. For such a small finite element system like Figure 18, where the half bandwidth is 12 and the total D.O.F equal to 30, the percentage of CPU associated with equation solving is negligible (0.5 %). Most of the times are spent on effective stiffness matrix, nonlinear force vectors generation and I/O. These parts have been discussed at previous chapter and it is believed that high parallel efficiency can be achieved. For a large half bandwidth finite element problem such as a half bandwidth equal to 100, the equation solving might become the dominant CPU time-comsuming, however, the Cholesky decomposition manifest a good speed up using a suitable amount of processors. From above investigations FENRIS appears well suited for parallel implementations.

8. Summary

This is to summarize the status of the work of this research. The main concerns of this research are to (1) develop a parallel finite element approach for crash dynamics analysis, (2) incorporate the parallel approach into a commercial finite element system, and (3) evaluate the performance of the methods with the GM S-shape test problem. In summary, work to date has progressed well on several fronts toward the research objectives. A baseline software system, good parallel crash dynamics methods, and validation results on the test problem have all been achieved. The initial results are encouraging and indicate that the key computation steps in crash analysis can benefit significantly from parallel computers. More detailed studies should be carried out on realistic problems with all major computation step being implemented in parallel.

Work is now proceeding to investigate a parallel element generation and nonlinear force vectors capability and evaluate the results on the S-Frame test problem. A recent meeting was held with CRAY Research representative who expressed interest in cooperative efforts to migrate FENRIS to the CRAY X-MP for parallel testing.

References

1. Butler, T., and Michel, D. NASTRAN, A Summary of Functions and Capabilities of the NASA Structural Analysis Computer System. NASA SP - 260 (1971).

2. Fredriksson, B., and Mackerle, J. Partial List of Major Finite Element Programs and Description of Some of Their Capabilities. In State-of-the-Art Surveys on Finite Element Technology (Noor and Pilkey, Editors). ASME Publication H00290, (1983) pp. 363-403.

3. Noor, A., Storaasli, O., and Fulton, R. Finite Element Technology in the Future. Impact of New Computations on Computational Mechanics. (Noor and Pilkey, Editors), ASME Special Publication H00275 (November 1983) pp. 1 - 32.

4. Storaasli, O., Peeples, S., Crockett, T., Knott, J., and Adams, L. The Finite Element Machine: An Experiement in Parallel Processing. NASA TM 84514 (1982).

5. Matelan, N., The FLEX/32 Multicomputing Enviroment. In Research in Structure and Dynamics--1984. NASA CP 2335 (October 1984) pp. 1 - 14.

6. Storaasli, O., Ransom, J., and Fulton, R. Structural Dynamic Analysis on a Parallel Computer: The Finite Element Machine. AIAA Paper 84- 0966-CP, presented at 25th AIAA/ASME/AHS Structures, Structural Dynamics and Materials Conference, Palm Springs, CA (14-16 May 1984)

7. Ransom, J., Storaasli, O., and Fulton, R. Application of Concurrent Processing to Structural Dynamic Response Computations. In Research in Structures and Dynamics--1984, NASA CP 2335 (Oct. 1984) pp. 31-44

8. Bostic, S., and Fulton, R. E., A Concurrent Processing Approach to Structural Vibration Analysis. 26th AIAA/ASME/ASCE/AHA Structures, Structural Dynamics and Materials Conference, Orlando, FL (15-17 April 1985).

9. Adams, L., and Ortega, J. A Multicolor SOR Method for Parallel Computation. Proc. 1982 Int. Conf. Parallel Processing, pp. 53-56.

10. Adams, L., and Jordon, H. Is SOR Color-Blind? ICASE Resport 84-14, NASA - Langley Research Center (1984).

11. Johnson, O., Micchelli, C., and Paul, G. Polynomial Preconditioners for Conjugate Gradient Calculations. SIAM J. Num. Anal. 20 (1983) pp. 362 - 376.

12. Adams, L. Iterative Algorithms for Large Sparse Linear System on Parallel Computers. Ph.D. Thesis, University of Virginia (1982).

13. Adams, L. An M-Step Preconditioned Conjugate Gradient Method for Parallel Computation. Proc. 1983 Int. Conf. Parallel Processing, pp. 36 - 43.

14. Adams, L. M-Step Preconditioned Conjugate Gradient Methods. To Appear in SIAM J. Sci. Stat. Comp.

15. Schreiber, R., and Tang, W. Vectorizing the Conjugate Gradient Method. Proc. Symposium Cyber 200 Applications, Ft. Collins, CO (1982)

16. Poole, E., and Ortega, J. Incomplete Cholesky Conjugate Gradient on the Cyber 203/205. Proc. 1984 Cyber 205 Applications Seminar.

17. Bostic, Susan W., and Fulton, R. Implementation of the Lanczos Method for Structural Vibration Analysis on a parallel Computer. AIAA Paper No. 86 - 0930 - CP, AIAA/ASME/ASCE/AHS 27th Structures, Structural Dynamics and Materials Conference, San Antonio, TX (19-21 May 1986).

18. Fulton, R. E. The Impact of Parallel Computing on Finite Element Computations. Presented at International Conference on Reliability of Methods for Engineering Analysis, Swansea, U.K. (9-11 July 1986).

19. George, A., Heath, M.T., and Liu, J., Parallel Cholesky Factorization on a Shared-Memory Multiprocessor. Tech Report ORNL-6124, Oak Ridge National Laboratory, March 1985.

20. Newmark, N. A Method of Computation for Structural Dynamics. J. Eng. Mech. Div., ASCE (July 1959) EM3, pp. 67-94.

21. Goehlich, D., Komzsik, L., and Fulton, R. E., Decomposition of Finite Element Matrices on Parallel Computers. To be presented at ASME Computers in Engineering Conference, August 10-14, 1987, N.Y.

22. Geist, A., Solving Finite Element Problems With Parallel Multifrontal Schemes, Second Conference on Hypercube Multiprocessors, September 29 - October 1, 1986, Knoxville, Tennessee.

23. Ni, Chi-Mou Impact Response of Curved Box Beam-Column with Large Global and Local Deformations. AIAA Paper No. 73 - 401, AIAA/ASME /SAE 14th Structures, Structural Dynamics, and Materials Conference, Williamsburg, Virginia/March 20-22, 1973.

24. Fulton, R. E., and Chiang, K. N., Computational Crash Dynamics Methods for Fifth Generation Supercomputers. GM Project Final Reports Phase I, Dec. 1986.

25. Chiang, K. N., and Fulton, R. E., Nonlinear Dynamics Methods for Parallel Computers. To be presented at ASCE Fifth Conference on Computing in Civil Engineering, March 29-31, 1988, Alexandria VA.

26. Malone, J. G., Automated Mesh Decomposition and Concurrent Finite Element Analysis for Hypercube Multiprocessor Computers, G.M. labs. Report, Engineering Mechanics Department, Warren, MI. May 18, 1987.

27. Wilson, E. L., and Dovey, H. H., Solution or Reduction of Equilibrium Equations for Large Complex Structural Systems, Journal of Advances in Engineering Software, 1978 Vol. 1, No. 1 pp. 19-25.

28. Mondkar, D. P., and Powell G. H., Large Capacity Equation Solver for Structural Analysis, Journal of Computers & Structures, 1974. Vol. 4 pp. 699 - 728.

29. Lyzenga, G. A., Raefsky, A., and Hager, B. H., Finite Elements and the Conjugate Gradients on a Concurrent Processor, CalTech/JPL Report C3P-119, Dec. 1984.

30. Argyris, J., Balmer, H. A., Doltsinis, J. St., and Kurz, A., Computer Simulation of Crash Phenomena, International Journal of Numerical Methods in Engineering, Vol. 22, pp. 497-519, 1986.

31. Storaasli, O. O., and Bergan, P., A Nonlinear Substructure Method for Concurrent Processing Computers, AIAA Publ. 86-0852, 1986.

32. Nygard, M. K., The Free Formulation for Nonlinear Finite Elements with Applications to Shells, Report No. 86-2, Division of Structural Mechanics, The Norwegian Institute of Technology, Norway, Dec. 1986.

33. FENRIS (Finite Element NonlineaR Integrated System) System Manual, Theory - Program outline - Data Input, by NTH-SINTEF-VERITEC, Norway, 1987.

34. Bathe, K. J., Finite Element Procedures in Engineering Analysis, Prentice - Hall Inc., NJ, 1982.

35. Dhatt, G., and Touzot, G., The Finite Element Method Displayed, John Wiley & Sons Inc., NY. 1984.

36. Cook, R. D., Concepts and Applications of Finite Element Analysis, Second Edition, John Wiley & Sons Inc., NY, 1981.

37. Irons, B. M., A Frontal Solution Program for Finite Element Analysis, Int. J. Num. Meth. Engrg., Vol 2. pp. 5-32, 1970.

38. Ashcraft, C. C., Parallel Reduction Methods for the Solution of Banded Systems of Equations, Computer Science Dept., Research Publi. GMR - 5094, June 30, 1985., Warren, MI.

39. Ni, Chi-Mou, A General Purpose Analytical Technique and Program, 'NONDRIS', for Nonlinear Dynamic Response of Integrated Structures, E. M. Dept., Research Report EM-507, April 20, 1981., Warren, MI.

40. Intel iPSC System Overview Manual, Order No. 175278 - 001.

41. ANSYS Theoretical Manual, Swanson Analysis System Inc. 3rd Edition, March 1, 1986.

42. Abu-Shumays, I. K., Comparison of Methods and Algorithms for Tridiagonal Systems and for Vectorization of Diffusion Computations. Bettis Atomic Power Lab., DE-AC11-76PN00011, W. Mifflin, PA.

43. Kershaw, D. Solution of Single Tridiagonal Linear Systems and Vectorization of The ICCG Algorithm on the CRAY-1, Lawrrence Livermore National Lab., ISBN 0-12-592101-2, Livermore, CA. 1982.

44. Clough, R. W. and Penzien, J., Dynamics of Structures, McGraw-Hill, 1975.

45. Katona, M. G. and Zienkiewicz, O. C., A Unified Set of Single Step Algorithms Part 3: The Beta-m Method, A Generalization of the Newmark Scheme, Int. J. Num. Meth. Engrg. Vol 21. pp 1345-1359, 1985

46. D'Souza, A. F. and Garg, V. K., Advanced Dynamics Modeling and Analysis, Printice-Hall Inc., NJ, pp 198-228, 1984.

47. Duff, I. S. and Reid, J. K., The Multifrontal Solution of Indefinite Sparse Symmetric Linear Systems, ACM Trans. on Math. Software, Vol.9 pp. 302-325

48. Hilton, E. and Owen, D. R. J., Finite Element Programming, Academic Press. NY, 1979.

49. Bryson, J. W., ORVIRT.PC: A 2-D Finite Element Fracture Analysis Program for a Microcomputer,, ORNL-6208, Oct. 1985.

50. Brebbia, C. A., Finite Element System, A Computational Mechanics Publication, NY, 1985.

51. Parlett, B. N., Nour-Omid, B., Implementation of Lanczos Algorithms on vector computers, N00014-76-C-0013, UC Berkeley, CA, 1985.

52. Hallquist, J. O., Theoretical Manual for DYNA3D, Lawrence Livermore Lab., CA, March 1983.

53. O'Leary, P. O., Parallel Implementation of the Block Conjugate Gradient Algorithm, Journal of Parallel Computing, pp 127-139, 1987.

54. Henk A. van der Vorst, Large Tridiagonal and Block Tridiagonal Linear System on Vector and Parallel Computers, Journal of Parallel Computing, pp 45-54, 1987.

55. Delves, L. M. and Sanba, A. S., Band Matrices on the DAP, GR/B/24332, University of Liverpool and University of Birmingham.

56. Meurant, G., Multitasking the Conjugate Gradient Method on the CRAY X-MP/48, Parallel Computing, Vol. 5, pp 267-280, 1987.

57. Benson, D.J., Hallquist, J.O. and Stillman, D.W., DYNA3D, INGRID and TAURUS - An Integrated, Interactive Software System for Crashworthiness Engineering, Lawrence Livermore Lab. CA, W-7405-Eng-48.

58. Bostic, S. W. and Fulton R. E., Experience with the Lanczos Method on a Parallc Computer, ASME Computers in Engineering Conference, August 9-13, 1987, N.Y.

59. Parkinson, D., Organizational Aspects of Using Parallel Computers, Parallel Computing Vol. 5, pp 75-83, 1987.

60. Hockney, R. W., Parametrization of Computer Performance, Parallel Computing Vol, 5, pp 97-103, 1987.

61. "FLEX/32 Multicomputer - System Overview", Document 030-000-001, Flexible Computer Corporation, Dallas, Texas, 1985.

APPENDIX

A. FENRIS - A General Purpose Nonlinear Finite Element Program

FENRIS is a large scale, general program for nonlinear finite element analysis. The program name is an abbreviation for Finite Element NonlineaR Integrated System. The development of FENRIS started in 1980 as a project between the Norwegian Institute of Technology (NTH), The Society for Industrial and Technical Research (SINTEF) and The Norwegian VERITAS. The main intention of the project has been to develop a new type of "truly nonlinear" program package. The program, as well as the technique of data storage, has been designed to correspond with the theoretical basis and major computational steps of nonlinear analyses. Another design concept for FENRIS has been to construct the program as a highly modular building kit. This has been done through isolating the various computational steps as much as possible from each other in the program software. This building block philosophy makes it easy to extend the program step by step and to readily accomodate new programming staff during development.

• Dynamic Algorithms

For dynamic problems, the body forces also contain the inertia force from D'Alembert's principle; the body force can be written as (Ref.32)

$$R^E = R_{int} + R^D + R^I \tag{A.1}$$

The subscript E, int, D and I denote external forces, internal forces, damping forces, and inertia forces, relatively. R^D may be used on the form

$$R^D = C\frac{du}{dt} \tag{A.2}$$

where C is a damping coefficient. The program also includes capabilities for other types of damping. R^I is obtained from the D'Alembert's principle as

$$R^I = \rho\frac{d^2u}{d^2t} \tag{A.3}$$

where ρ is density. For time step i, eq.(A.1) may be written as

$$R_i^I + R_i^D + R_{int,i} = R_i^E \tag{A.4}$$

The internal reaction force vectors are computed element by element then transformed from the local co-rotated element coordinate system (ref.32) to the global reference system and assembled for all elements. The inertia force vector and the damping force vector are also in reality computed element-wise and assembled with use of the global matrices M and C. The incremental form of the governing equation may be written as

$$M_i\Delta\ddot{u} + C_i\Delta\dot{u} + K_{I,i}\Delta u = R_i^E - (R_{i-1}^I + R_{i-1}^D + R_{int,i-1}) \tag{A.5}$$

M_i is the mass matrix. C_i the incremental damping matrix and $K_{I,i}$ the incremental stiffness matrix. $K_{I,i}$ may be taken as the tangential element stiffness K_T,assembled to a global system matrix, or, the K_I may be used for several consecutive steps. $\Delta u_i, \Delta\dot{u}_i, \Delta\ddot{u}_i$ are the incremental displacement, velocity, and acceleration vectors. The Δ denotes finite but "small" increments corresponding to the difference between the two states considered. Equation (A.5) is used in connection with load increments and equilibrium iterations.

The program assumes that the Rayleigh damping matrix (ref. 44) is constructed from the following form

$$C = \alpha_1 M + \alpha_2 K_T + C_V \tag{A.6}$$

where α_1, α_2 are damping factors. and C_v is a diagonal matrix representing viscous dashpots prescribed at nodes.

• Implicit Time Integration Algorithm

A general incremental-iterative scheme combined with the Newmark family of implicit time integration operators is used to solve the dynamic equations. The basic assumptions of the Newmark operators are (ref.20)

$$\dot{u}_{i+1} = \dot{u}_i + (1 - \gamma)\ddot{u}_i h + \gamma \ddot{u}_{i+1} h \tag{A.7}$$

$$u_{i+1} = u_i + \dot{u}_i h + (\frac{1}{2} - \beta)\ddot{u}_i h^2 + \beta \ddot{u}_{i+1} h^2 \tag{A.8}$$

Here h is the incremental time step

$$h = \Delta t = t_{i+1} - t_i$$

which may vary throughout the analysis. From equations (A.7) and (A.8) the corresponding increments of velocities and accelerations are

$$\Delta \dot{u}_{i+1} = \frac{\gamma}{\beta h} \Delta u_{i+1} - \frac{\gamma}{\beta} \dot{u}_i - (\frac{\gamma}{2\beta} - 1) h \ddot{u}_i \tag{A.9}$$

$$\Delta \ddot{u}_{i+1} = \frac{1}{\beta h^2} \Delta u_{i+1} - \frac{1}{\beta h} \dot{u}_i - \frac{1}{2\beta} \ddot{u}_i \tag{A.10}$$

Substitution equations (A.9) and (A.10) into equation (A.5) yields

$$\left[M\frac{1}{\beta h^2} + C\frac{\gamma}{\beta h} + K \right]\Delta u_{i+1} = R^E_{i+1} - (R^I_i + R^D_i + R_{int,i}) + M\left[\frac{1}{\beta h}\dot{u}_i + \frac{1}{2\beta}\ddot{u}_i \right] + \mathbf{C}\left[\frac{\gamma}{\beta}\dot{u}_i + (\frac{\gamma}{2\beta} - 1)h\ddot{u}_i \right] \tag{A.11}$$

By combining equations (A.2), (A.3), (A.6) and (A.11) the incremental - corrective equations become

$$\hat{K}\Delta\hat{u}_{i+1} = \Delta\hat{R}_{i+1} \tag{A.12}$$

where the effective stiffness matrix $\hat{K}$ is

$$\hat{K} = a_o M + c_o C_V + K_T \tag{A.13}$$

with the following constants

$$a_o = \frac{1}{1 + \frac{\alpha_2\gamma}{\beta h}}(\frac{1}{\beta h^2} + \frac{\alpha_1\gamma}{\beta h}) \tag{A.14}$$

$$c_o = \frac{1}{1 + \frac{\alpha_2\gamma}{\beta h}}(\frac{\gamma}{\beta h})$$

Here α_1 and α_2 are proportionality factors for the Rayleigh damping matrix (see equation A.6). Note that the effective stiffness depends on the time step h. The increment of the effective load is

$$\Delta\hat{R}_{i+1} = R^E_{i+1} - R_{int,i} + M[\hat{a}_i + \hat{b}_i(\alpha_1 - a_o\alpha_2) - \ddot{u}_i] + C_V\hat{b}_i(1 - c_o\alpha_2) \tag{A.15}$$

where

$$\hat{b}_i = (\frac{\gamma}{\beta} - 1)\dot{u}_i + (\frac{\gamma}{2\beta} - 1)h\ddot{u}_i \tag{A.16}$$

$$\hat{a}_i = \frac{1}{\beta h}\dot{u}_i + \frac{1}{2\beta}\ddot{u}_i \tag{A.17}$$

The set of equations is solved with respect to the effective displacement increment $\Delta\hat{u}_{i+1}$. The increment of the real displacement vector is the

$$\Delta u_{i+1} = \frac{1}{1 + \frac{\alpha_2\gamma}{\beta h}}(\Delta\hat{u}_{i+1} + \alpha_2\hat{b}_i) \tag{A.18}$$

and $\Delta\dot{u}_{i+1}, \Delta\ddot{u}_{i+1}$ can be obtained from equation (A.9), (A.10). The corresponding total vectors are

$$u_{i+1} = u_i + \Delta u_{i+1}$$

$$\dot{u}_{i+1} = \dot{u}_i + \Delta\dot{u}_{i+1} \tag{A.19}$$

$$\ddot{u}_{i+1} = \ddot{u}_i + \Delta\ddot{u}_{i+1}$$

B. Nonlinear Analyses on ANSYS

The nonlinear options in ANSYS include geometric nonlinearities, (large deflection, large rotation, and stress stiffening), special nonlinear elements, and nonlinear material behavior. Large deflection analysis is available for both static and dynamic problems. Geometry modifications are made at the end of each load increment; stress stiffening and large rotation effects are available for most libraries.

The nonlinear transient dynamic analysis is an extension of the static analysis that solves for the dynamic response of a structure under the action of applied time dependent loads. Inertia and damping effects may be included as well as all nonlinearities shown with the static analysis. The basic equation being solved in the nonlinear transient dynamic analysis is (ref.41)

$$M\ddot{u} + C\dot{u} + Ku = F(t) \tag{B.1}$$

where

F(t) = applied nodal force load vector

• Implicit Time Integration Algorithm

The equation is solved by an implicit direct integration scheme based on a modified Houbolt method. The method uses a cubic displacement function during an iteration. Thus, the velocity varies quadratically and acceleration varies linearly during the iteration. The displacement function is assumed as

$$u_t = a + b\ t + c\ t^2 + d\ t^3 \tag{B.2}$$

a set of four equations may be defined (at t = 0, h_1, h_2, h_3)

$$\begin{aligned} u_{i-3} &= a \\ u_{i-2} &= a + bh_1 + ch_1^2 + dh_1^3 \\ u_{i-1} &= a + bh_2 + ch_2^2 + dh_2^3 \\ u_i &= a + bh_3 + ch_3^2 + dh_3^3 \end{aligned} \tag{B.3}$$

where

$$\begin{aligned} h_1 &= t_{i-2} - t_{i-3} \\ h_2 &= t_{i-1} - t_{i-3} \\ h_3 &= t_i - t_{i-3} \end{aligned}$$

Solving equations (B.3) gives the four constants a, b, c, d as

$$\begin{aligned} a &= u_{i-3} \\ b &= e_1 u_i + e_2 u_{i-1} + e_3 u_{i-2} + e_4 u_{i-3} \\ c &= g_1 u_i + g_2 u_{i-1} + g_3 u_{i-2} + g_4 u_{i-3} \\ d &= k_1 u_i + k_2 u_{i-1} + k_3 u_{i-2} + k_4 u_{i-3} \end{aligned} \tag{B.4}$$

where

$$L = h_2^3 - h_1^3, \quad M = h_2^3 h_1 - h_1^3 h_2, \quad N = h_2^3 h_1^2 - h_1^3 h_2^2$$

$$P = h_3^3 - h_1^3, \quad Q = h_3^3 h_1 - h_1^3 h_3, \quad R = h_3^3 h_1^2 - h_1^3 h_3^2$$

$$\hat{e} = RM - NQ$$

$$e_1 = \frac{(Nh_1^3)}{\hat{e}}, \quad e_2 = \frac{-(Rh_1^3)}{\hat{e}}, \quad e_3 = \frac{(Rh_2^3 - Nh_3^3)}{\hat{e}}, \quad e_4 = \frac{(NP - RL)}{\hat{e}}$$

$$g_1 = \frac{-(Mh_1^3)}{\hat{e}}, \quad g_2 = \frac{(Qh_1^3)}{\hat{e}}, \quad g_3 = \frac{(Mh_3^3 - Qh_2^3)}{\hat{e}}, \quad g_4 = \frac{(QL - MP)}{\hat{e}}$$

$$k_1 = \frac{-(h_1 e_1 + h_1^2 g_1)}{h_1^3}, \qquad k_2 = \frac{-(h_1 e_2 + h_1^2 g_2)}{h_1^3}$$

$$k_3 = \frac{(1 - h_1 e_3 - h_1^2 g_3)}{h_1^3}, \qquad k_4 = \frac{-(1 + h_1 e_4 + h_1^2 g_4)}{h_1^3} \tag{B.5}$$

Differentiating equation (B.2) gives

$$\begin{aligned} \dot{u} &= b + 2\,c\,t + 3\,d\,t^2 \\ \ddot{u} &= 2\,c + 6\,d\,t \end{aligned} \tag{B.6}$$

By substituting equation (B.4) into equation (B.6), $\dot{u}$, $\ddot{u}$ can be rewritten as

$$\begin{aligned} \dot{u}_i &= (e_1 + 2g_1h_3 + 3k_1h_3^2)u_i + (e_2 + 2g_2h_3 + 3k_2h_3^2)u_{i-1} \\ &\quad + (e_3 + 2g_3h_3 + 2k_3h_3^2)u_{i-2} + (e_4 + 2g_4h_3 + 2k_4h_3^2)u_{i-3} \\ &= c_1\,u_i + c_3\,u_{i-1} + c_4\,u_{i-2} + c_5\,u_{i-3} \end{aligned}$$

$$\begin{aligned} \ddot{u}_i &= (2g_1 + 6h_3k_1)u_i + (2g_2 + 6h_3k_2)u_{i-1} + (2g_3 + 6h_3k_3)u_{i-2} \\ &\quad + (2g_4 + 6h_3k_4)u_{i-3} \\ &= c_2u_i + c_6u_{i-1} + c_7u_{i-2} + c_8u_{i-3} \end{aligned} \tag{B.7}$$

Combining equations (B.1) and (B.7) results in the governing dynamics equation in the following form

$$\hat{K}u_i = \hat{R} \tag{B.8}$$

The effective stiffness matrix $\hat{K}$ and the effective load vector $\hat{R}$ are

$$\begin{aligned} \hat{K} &= c_2\,M + c_1\,C + K \\ \hat{R} &= \Gamma(t_i) - M\,(c_6u_{i-1} + c_7u_{i-2} + c_8u_{i-3}) - C\,(c_3u_{i-1} + c_4u_{i-2} + c_5u_{i-3}) \end{aligned} \tag{B.9}$$

In the case of a constant time step size h, equation (B.7) reduce to the well known Houbolt algorithm (ref. 46).

$$\dot{u}_i = \frac{1}{6h}(11u_i - 18u_{i-1} + 9u_{i-2} - 2u_{i-3})$$
$$\ddot{u}_i = \frac{1}{h^2}(2u_i - 5u_{i-1} + 4u_{i-2} - u_{i-3}) \qquad \text{(B.10)}$$

where the effective stiffness matrix and load vector are

$$\hat{K} = \frac{2}{h^2} M + \frac{11}{6h} C + K$$
$$\hat{R} = F(t_i) + M\left(\frac{5}{h^2} u_{i-1} - \frac{4}{h^2} u_{i-2} + \frac{1}{h^2} u_{i-3}\right) \qquad \text{(B.11)}$$
$$+ C\left(\frac{3}{h} u_{i-1} - \frac{3}{2h} u_{i-2} + \frac{1}{3h} u_{i-3}\right)$$

For nonlinear transient analyses, the default of ANSYS extrapolates the displacements to the next time point for the stress solution if nonlinear materials are included. The displacement extrapolation is used to calculate the plastic strains which are, in turn, used to calculate the plastic load vector for the next time step. Extrapolation use the same form as the original assumption of cubic variation of displacements (equation B.2), but now the values are explicitly known. The last calculation relating to the nonlinear transient dynamic analysis iteration is to determine the iteration time step size to be used during the next iteration; this value of next time iteration step size must undergo various tests otherwise overshoot and unstable conditions might occur.

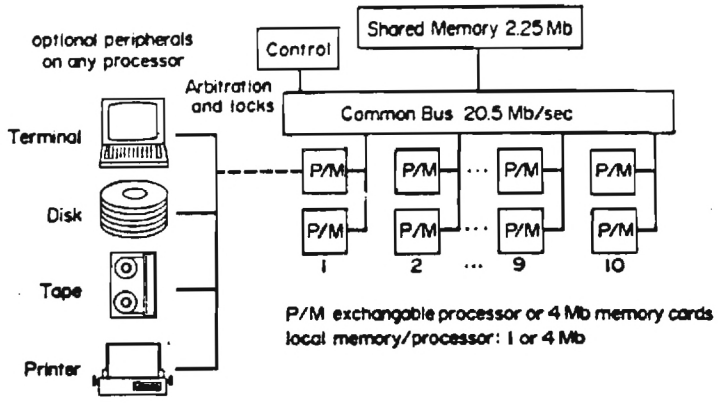


Figure 1. FLEX/32 Multicomputer.

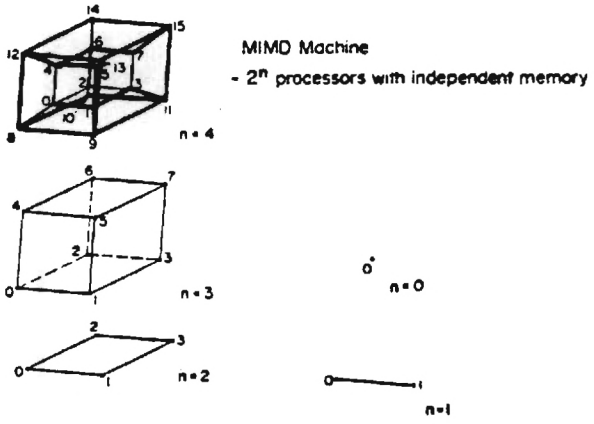


Figure 2. iPSC Hypercube Architecture.

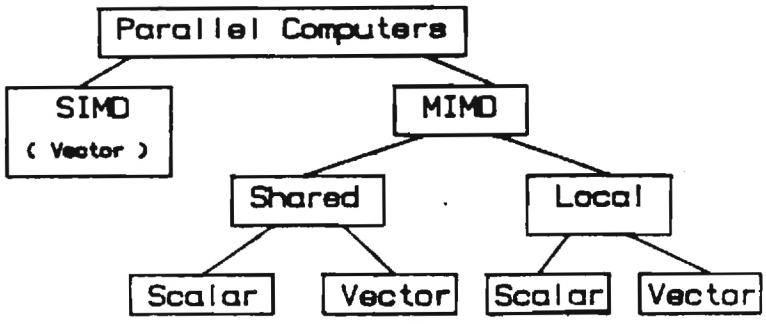


Figure 3. A Simple Taxonomy of Parallel Computers

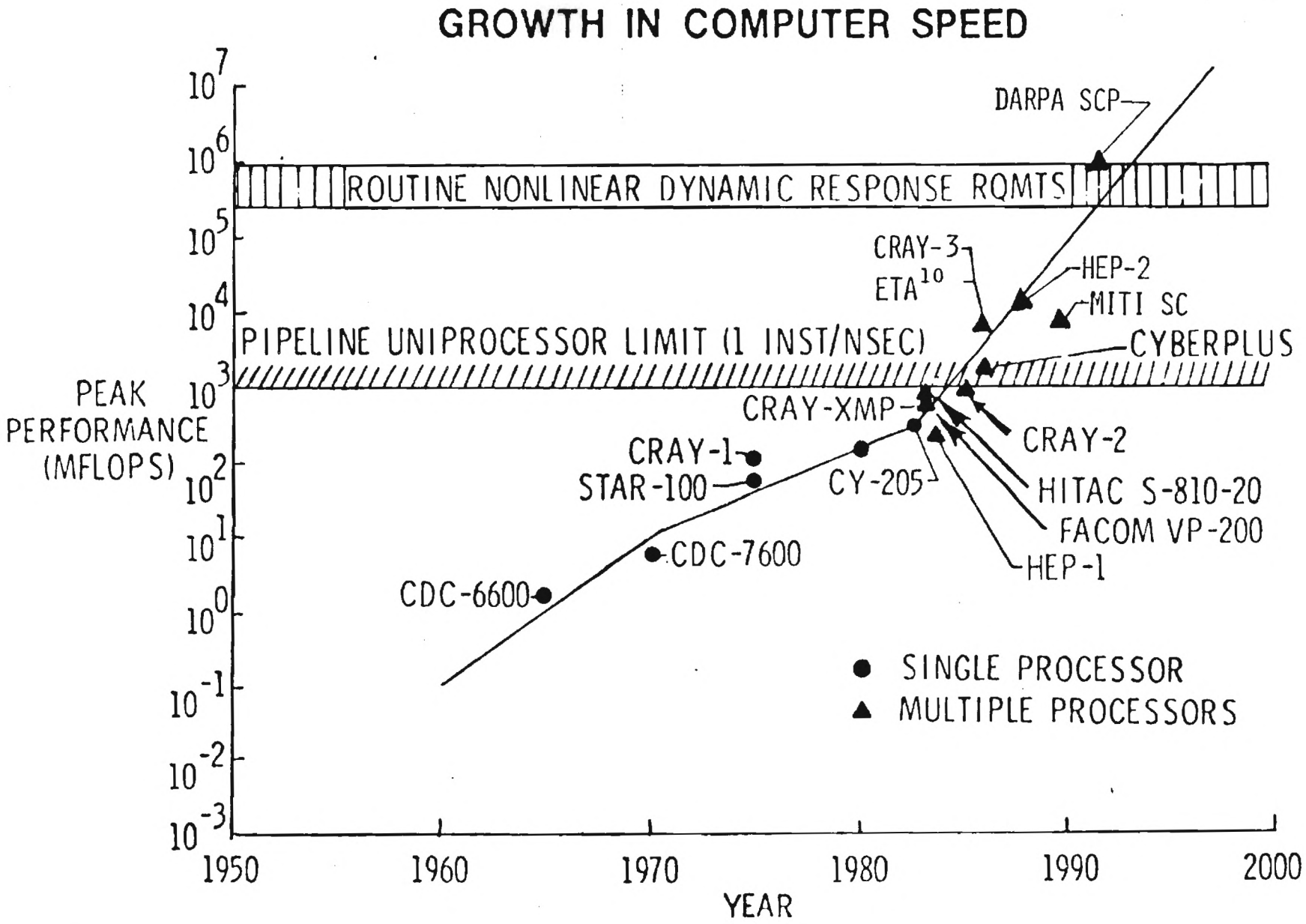


Figure 4. Growth in Computer Speed

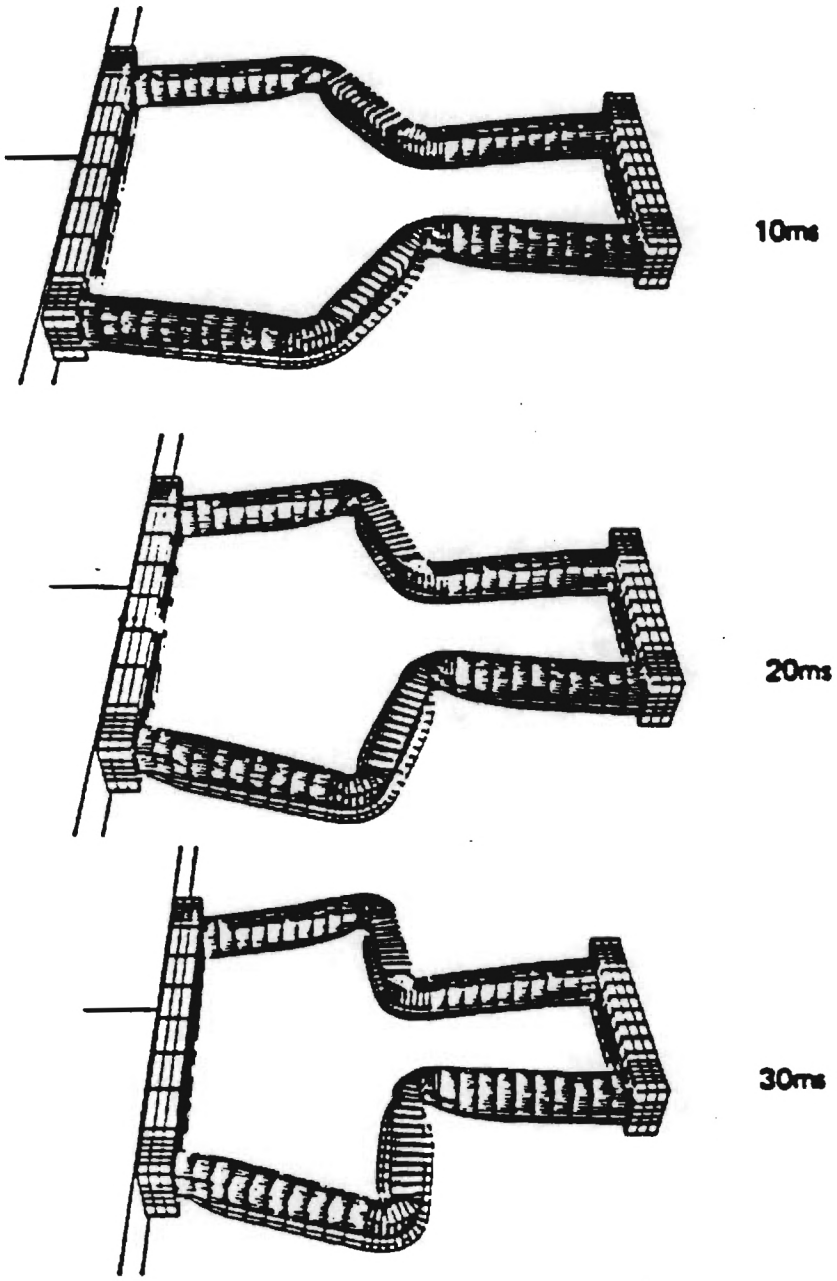
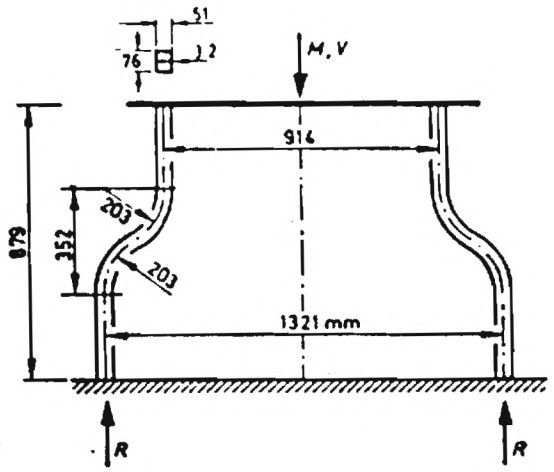


Figure 5. Deformed shapes at 10ms output intervals.



Impact mass $M = 1932$ kg
Impact velocity $V = 4.65$ m/s

Steel material

$E = 2.07 \cdot 10^5$ N/mm²

$\sigma_0 = 250$ N/mm²

$\frac{\sigma}{\sigma_0} = 1 + \left(\frac{\dot{\varepsilon}}{40}\right)^{0.2}$

$\rho = 7.9$ kg/dm³

Torque box subjected to impact

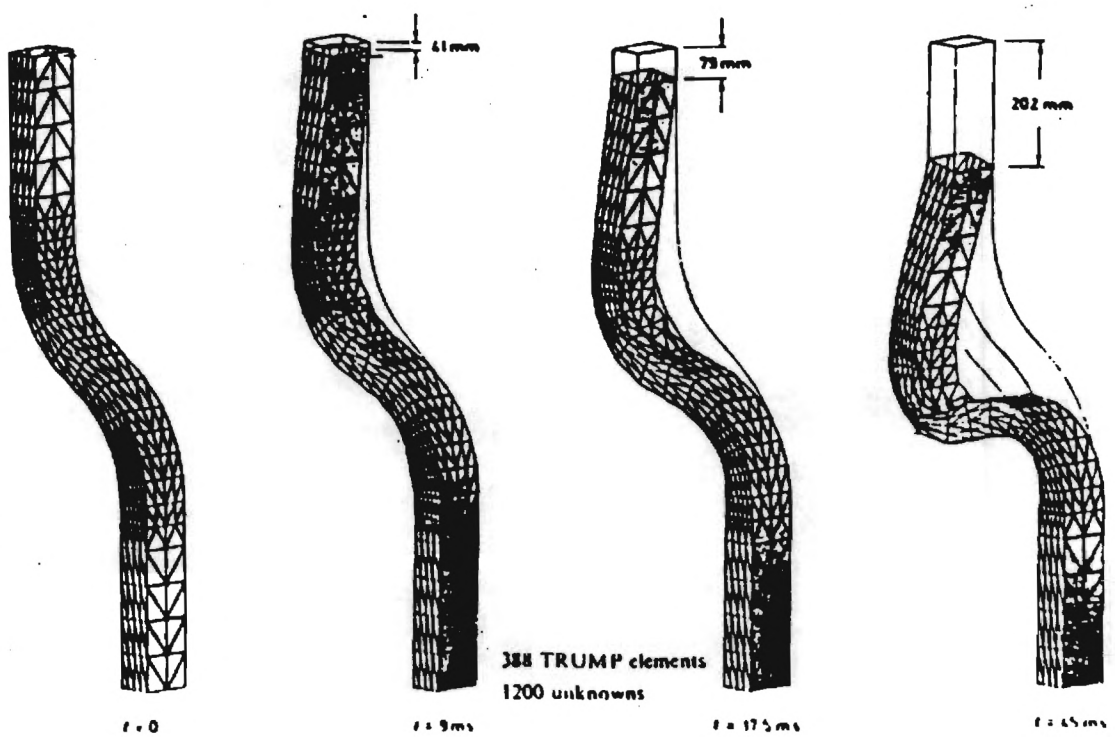


Figure 6. S-Frame Test Problem

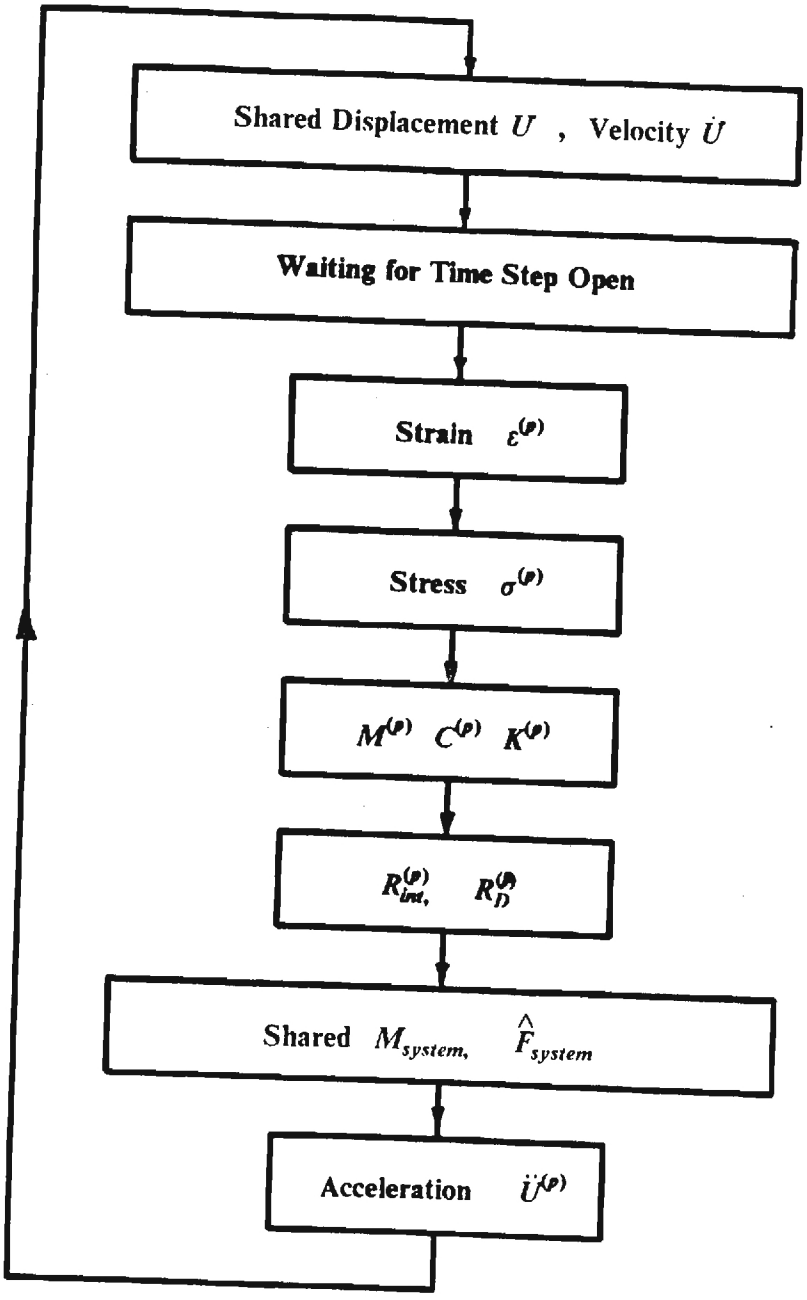


Figure 7. Concurrent Explicit Finite Element Method.

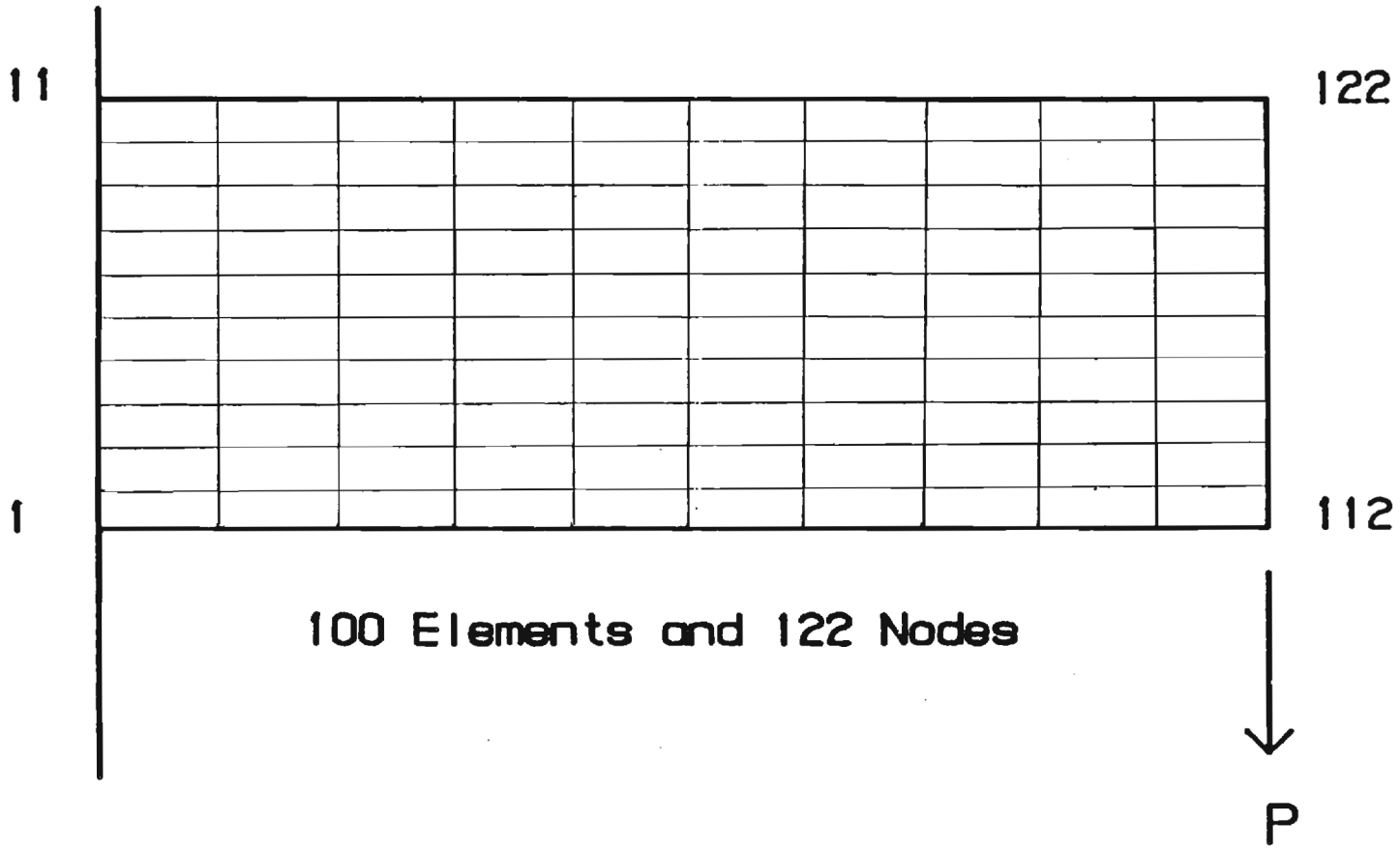


Figure 8. Cantilever Beam (FENRIS)

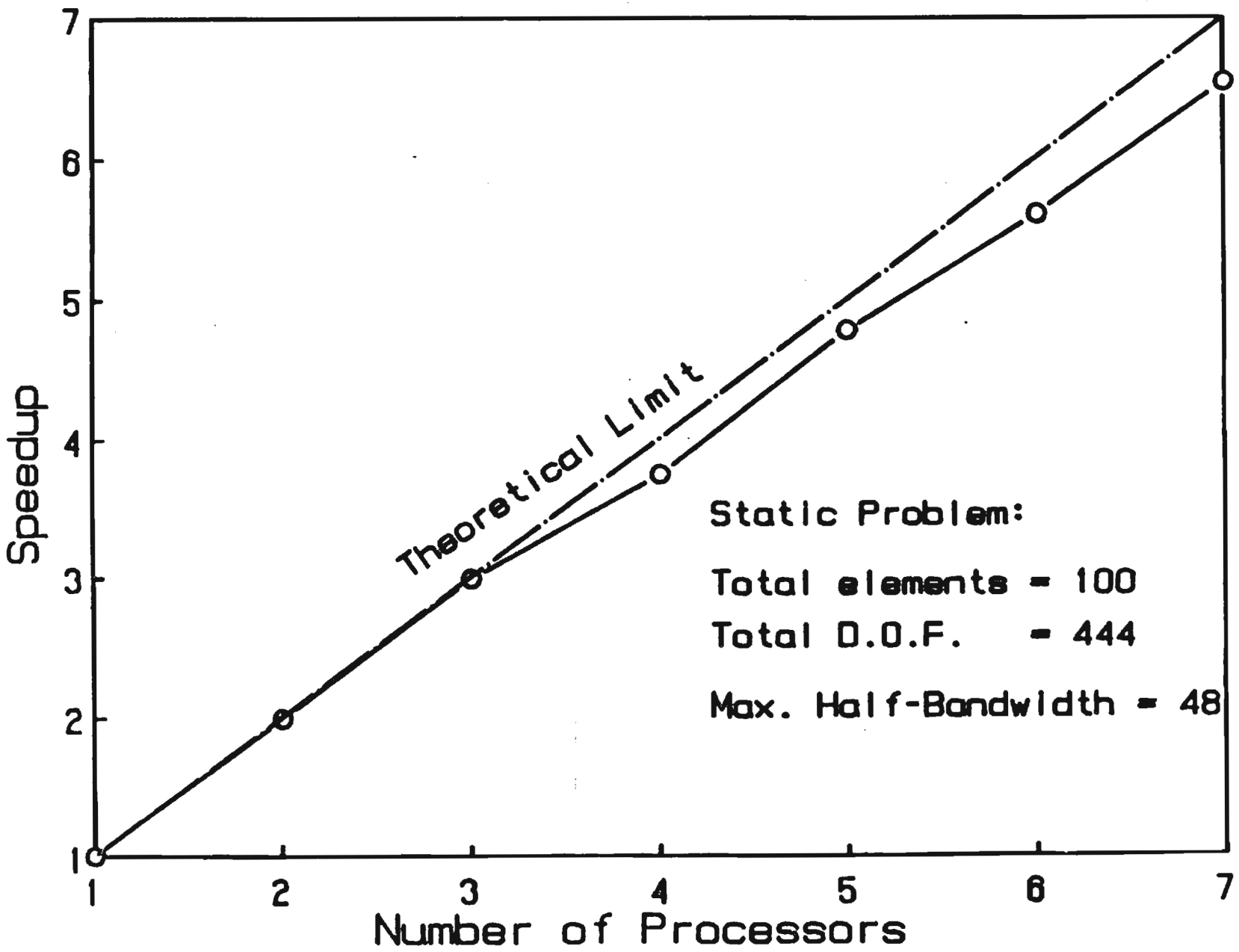


Figure 9. LDLT Decomposition (FLEX/32).

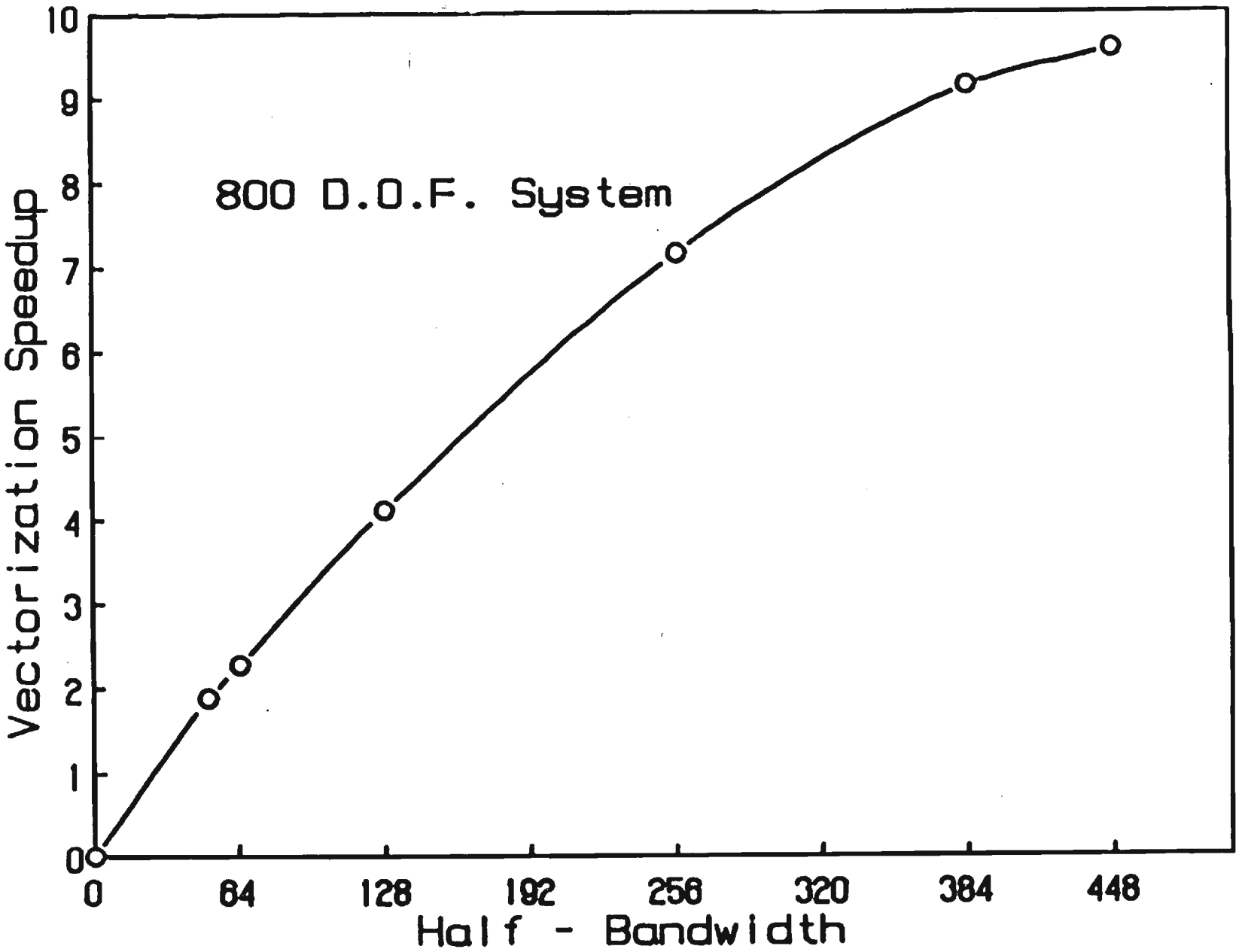
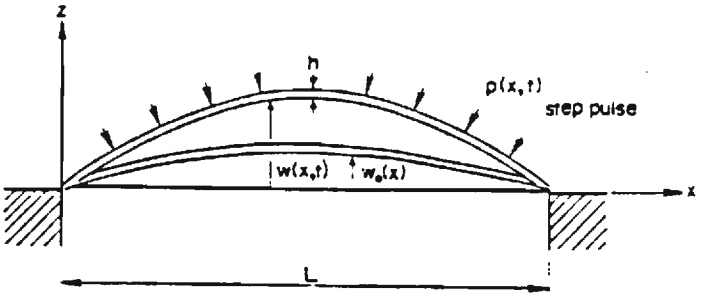


Figure 10. LDLT Decomposition (CRAY X-MP/48).



Geometry of a Shallow Arch.

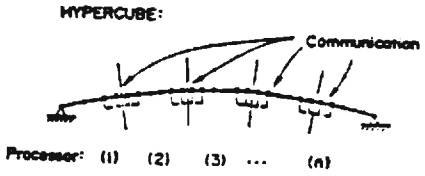


Figure 11. Intel iPSC
Communication Sequence.

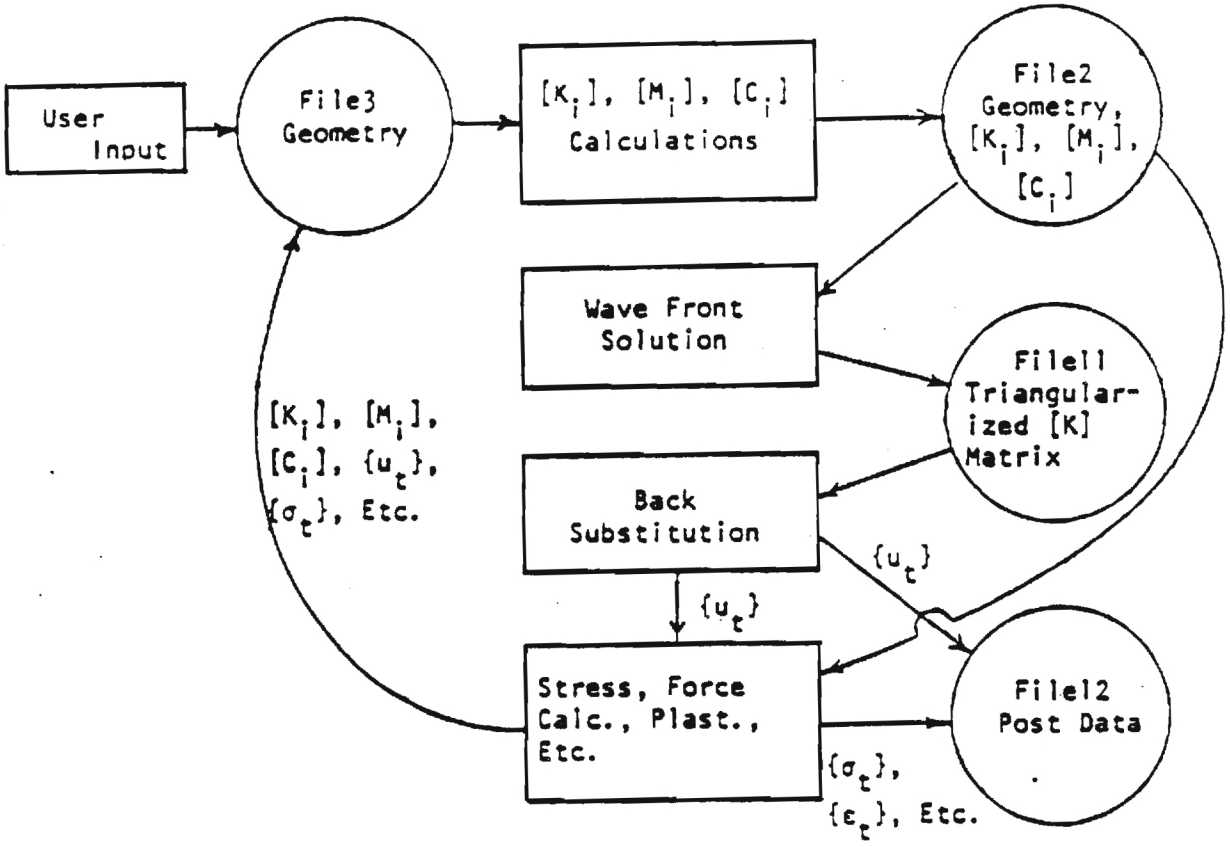


Figure 12. Nonlinear Transient Dynamic (ANSYS)

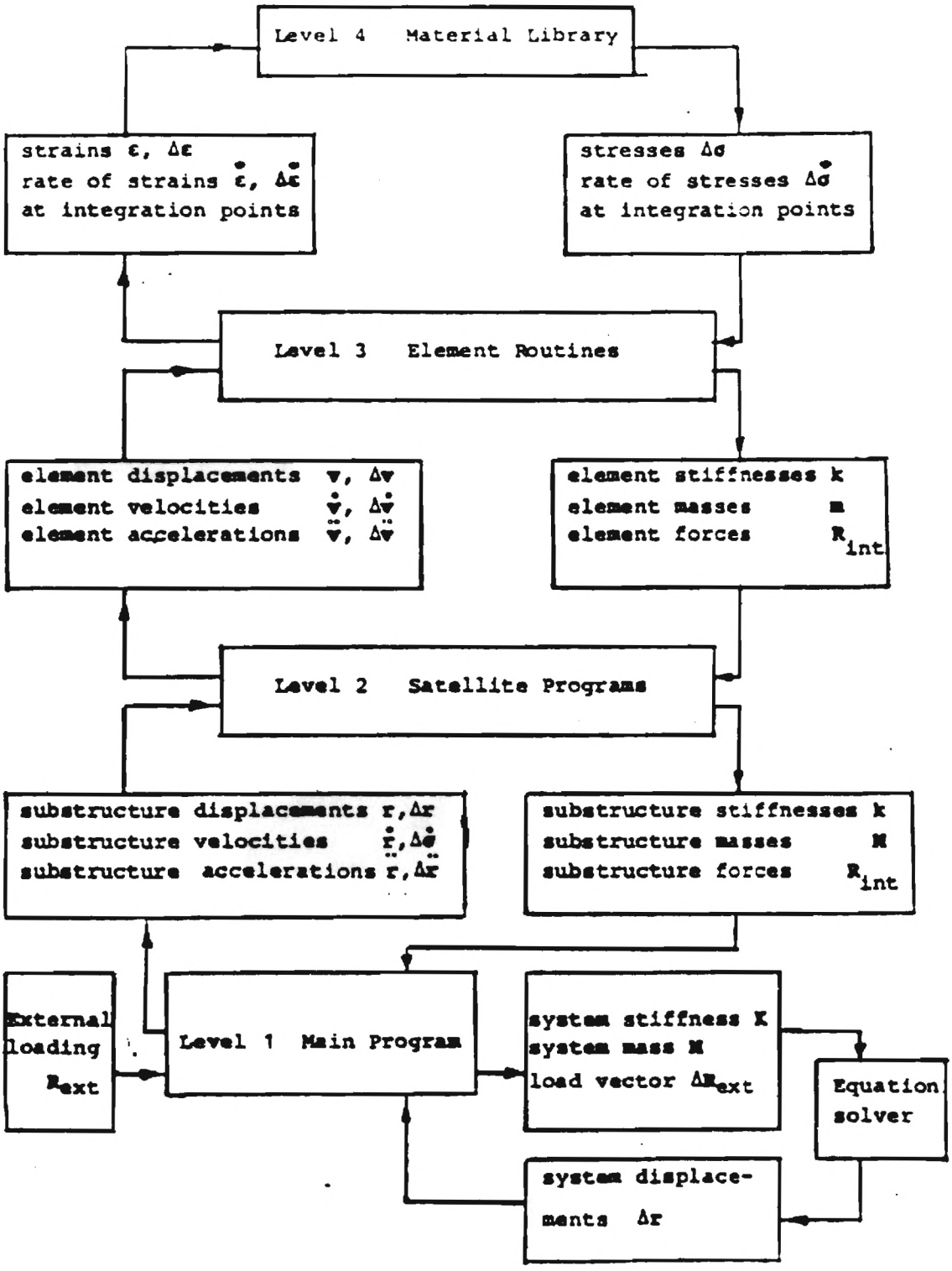


Figure 15. Nonlinear Analysis (FENRIS)

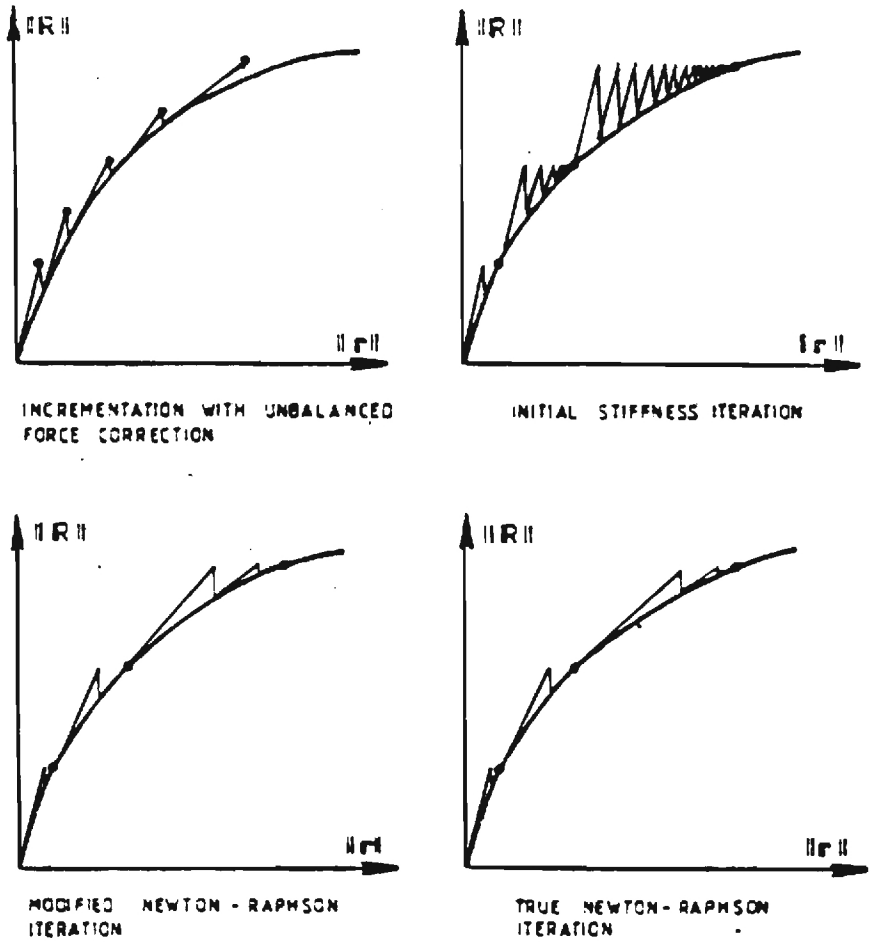


Figure 14. Illustration of the Alternative Solution Methods (FENRIS)

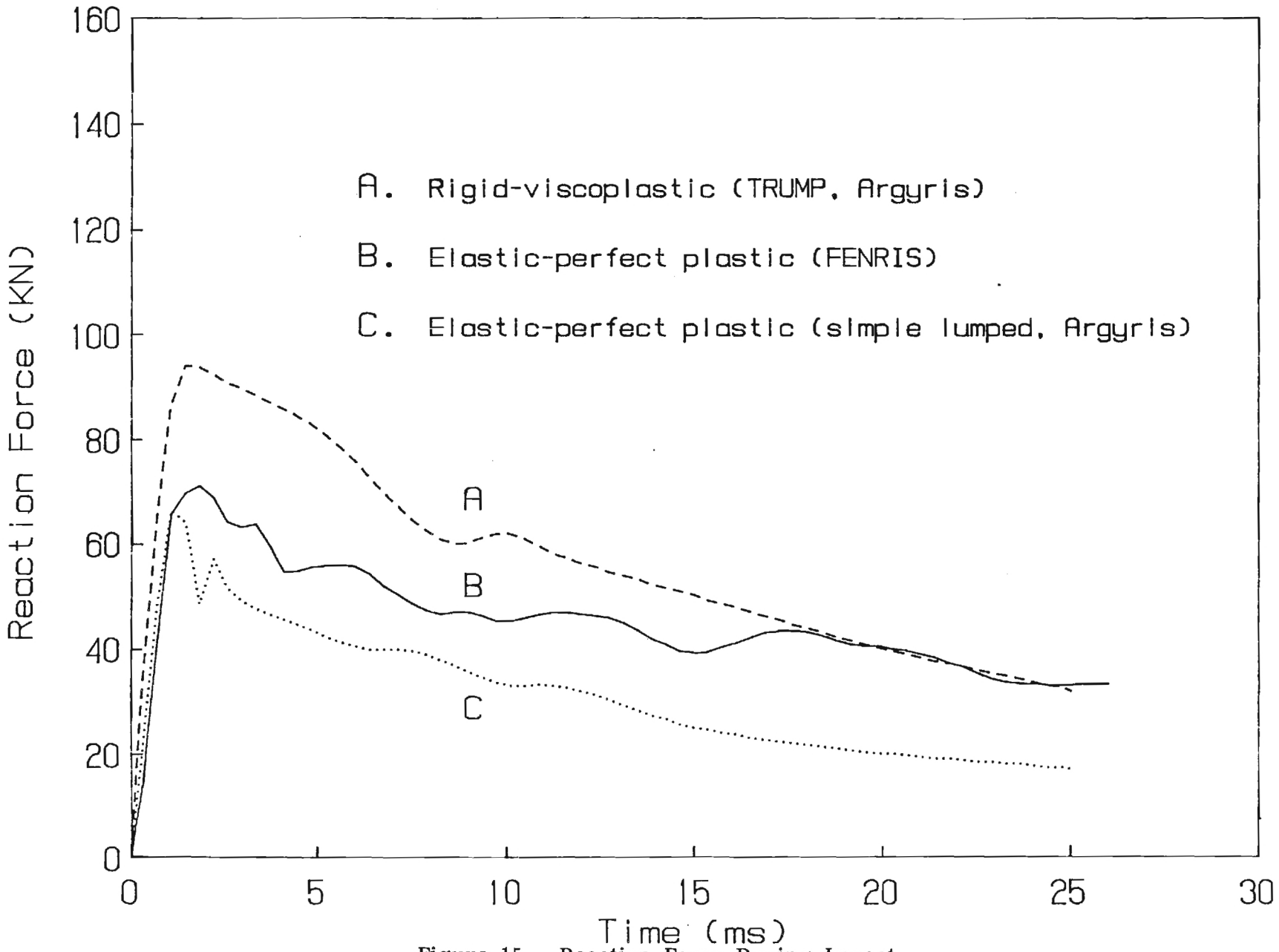


Figure 15. Reaction Force During Impact.

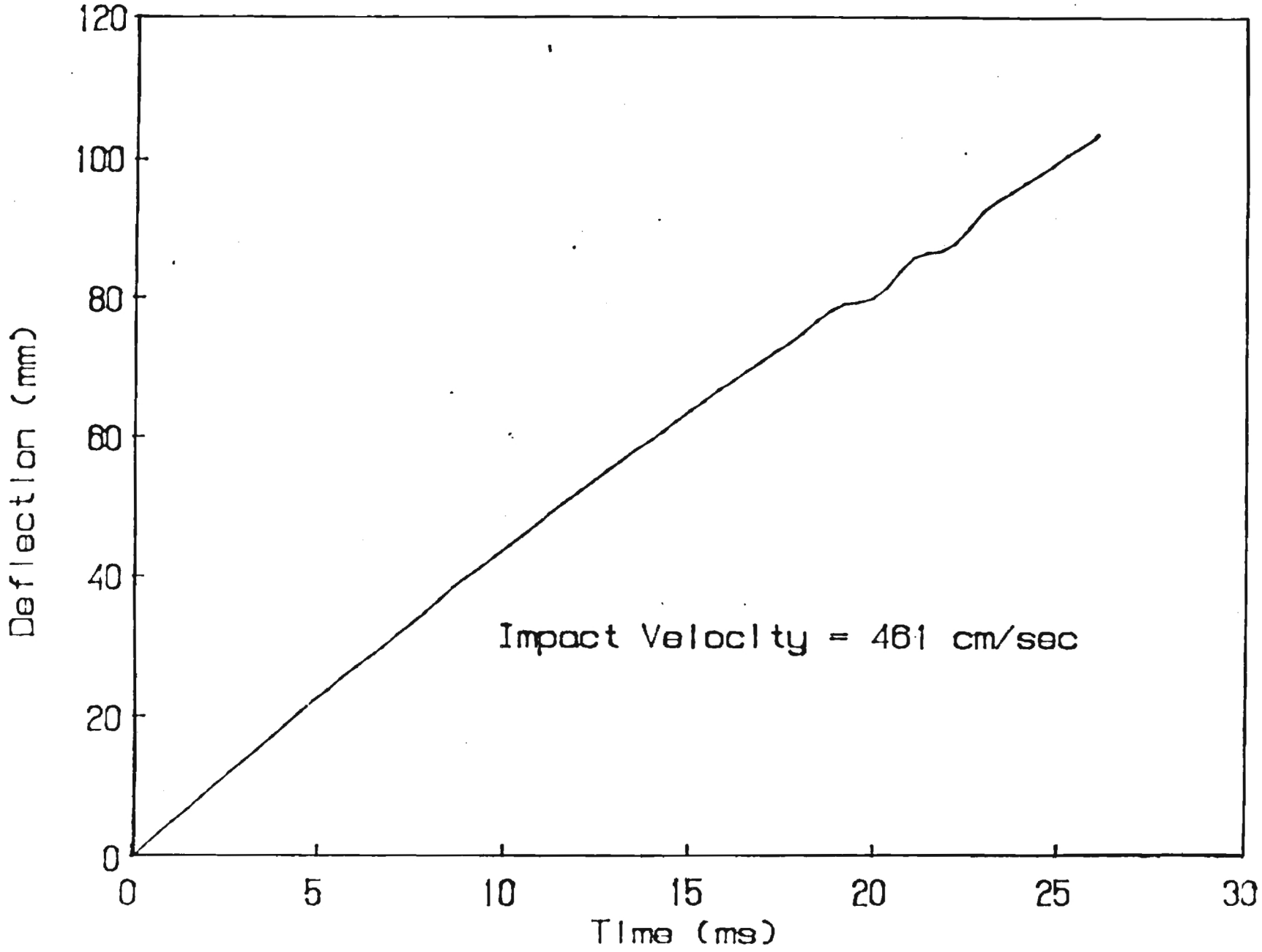


Figure 16. Time - Deflection.

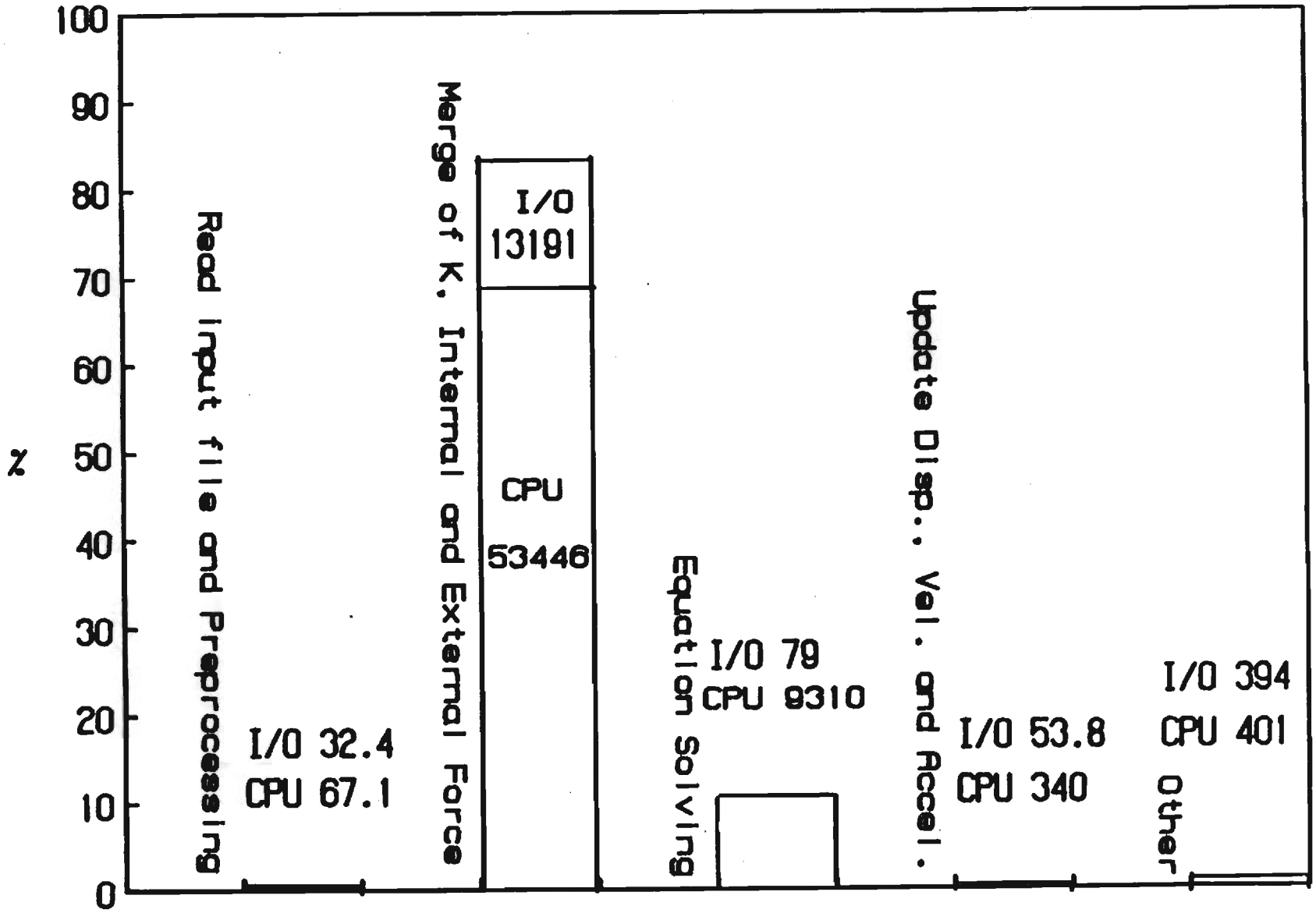


Figure 17. 449 D.O.F S-Frame.

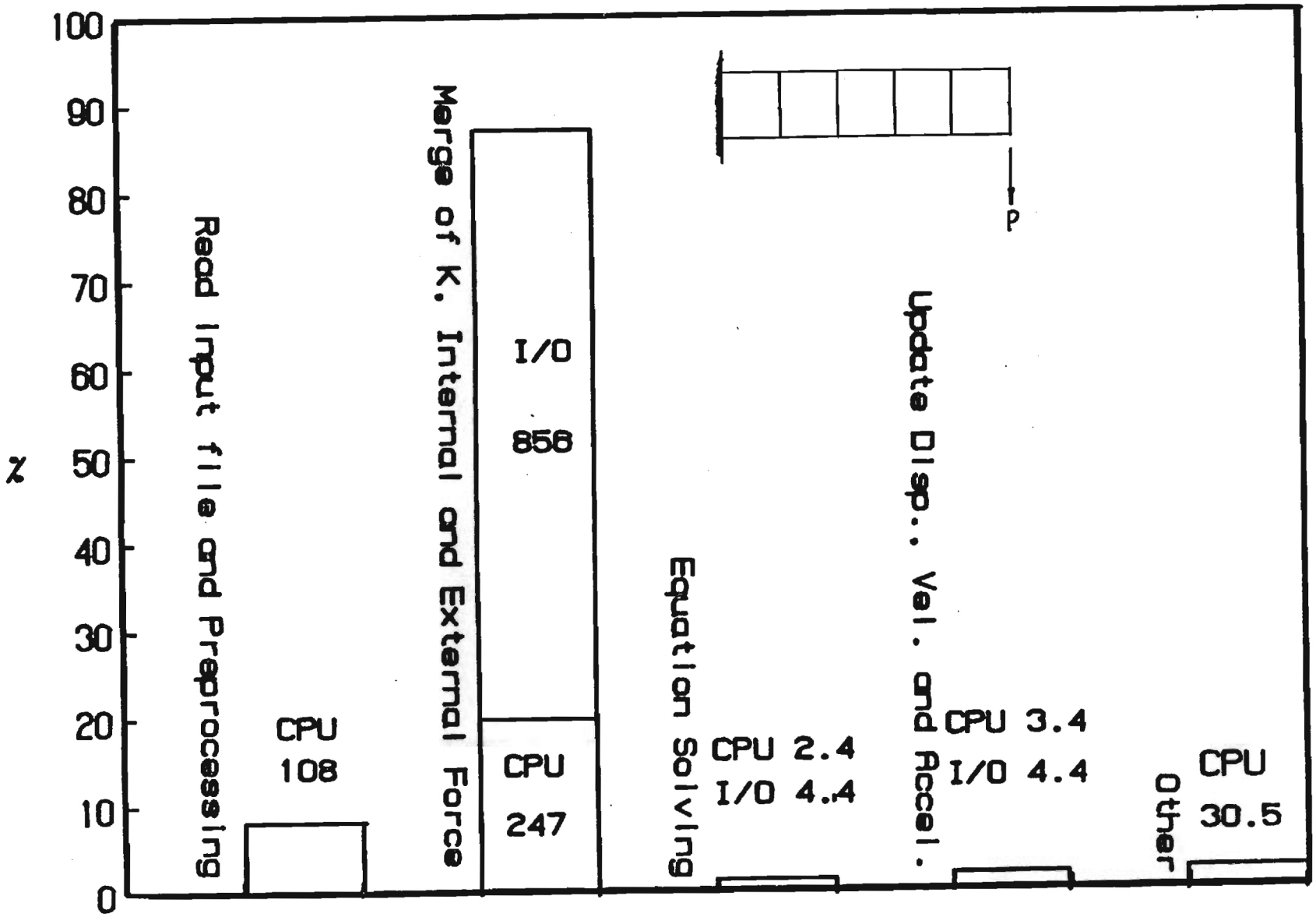


Figure 18. 30 D.O.F. Cantilever Beam.